Documentation d'IBM WebSphere DataPower XC10 Appliance version 2.5

Bienvenue dans la documentation d'IBM® WebSphere DataPower XC10 Appliance, dans laquelle vous trouverez des informations concernant l'installation, la gestion et l'utilisation de ce produit.

Initiation

Présentation générale d'IBM WebSphere DataPower XC10 Appliance
Démarrage rapide : Installation du matériel du dispositif
2.5+ Tutoriel : Démarrer avec des applications de grille de données simples
Nouveautés de la version 2.5
Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance
Mise à jour de WebSphere DataPower XC10 Appliance

Tâches courantes

Configuration de votre dispositif
Configuration des collectivités et des zones
Configuration des grilles de données
Configuration des clients
Administration des grilles de données
Développement d'applications pour accéder à des grilles de données simples
Surveillance
Sécurité

Traitement des incidents et support

Traitement des incidents
Portail d'assistance
Fix central
Forum
Notes sur l'édition
Page d'accueil du service de support logiciel IBM

Informations complémentaires

Formation à IBM WebSphere DataPower XC10 Appliance
Articles
Redbooks
Communauté dédiée au cache mémoire redimensionnable (cache élastique) d'IBM

Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Le produit sous licence IBM® WebSphere DataPower XC10 Appliance est un dispositif spécialement conçu et optimisé pour offrir une mise en cache simple, rapide et rentable pour les applications WebSphere.

Avantages

DataPower XC10 Appliance offre les avantages suivants :

Evolutivité en toute simplicité

DataPower XC10 Appliance contient une grille de données flexible de 240 Go que vous pouvez utiliser pour héberger les données de vos applications vitales. Pour ajouter davantage de mémoire à la grille de données, vous pouvez ajouter un autre dispositif à la configuration, créant ainsi une collectivité de dispositifs pour l'hébergement de vos données.

Utilisation simple et instantanée sans modification de code

Vous pouvez utiliser DataPower XC10 Appliance sans apporter de modification au code de vos applications. Vous pouvez intégrer directement le dispositif dans les scénarios suivants :

- Gestion de session pour les requêtes HTTP
- Prise en charge de la mémoire cache dynamique WebSphere Application Server

Tolérance aux pannes

Les données des grilles de données sont automatiquement répliquées afin de diminuer le risque de perte.

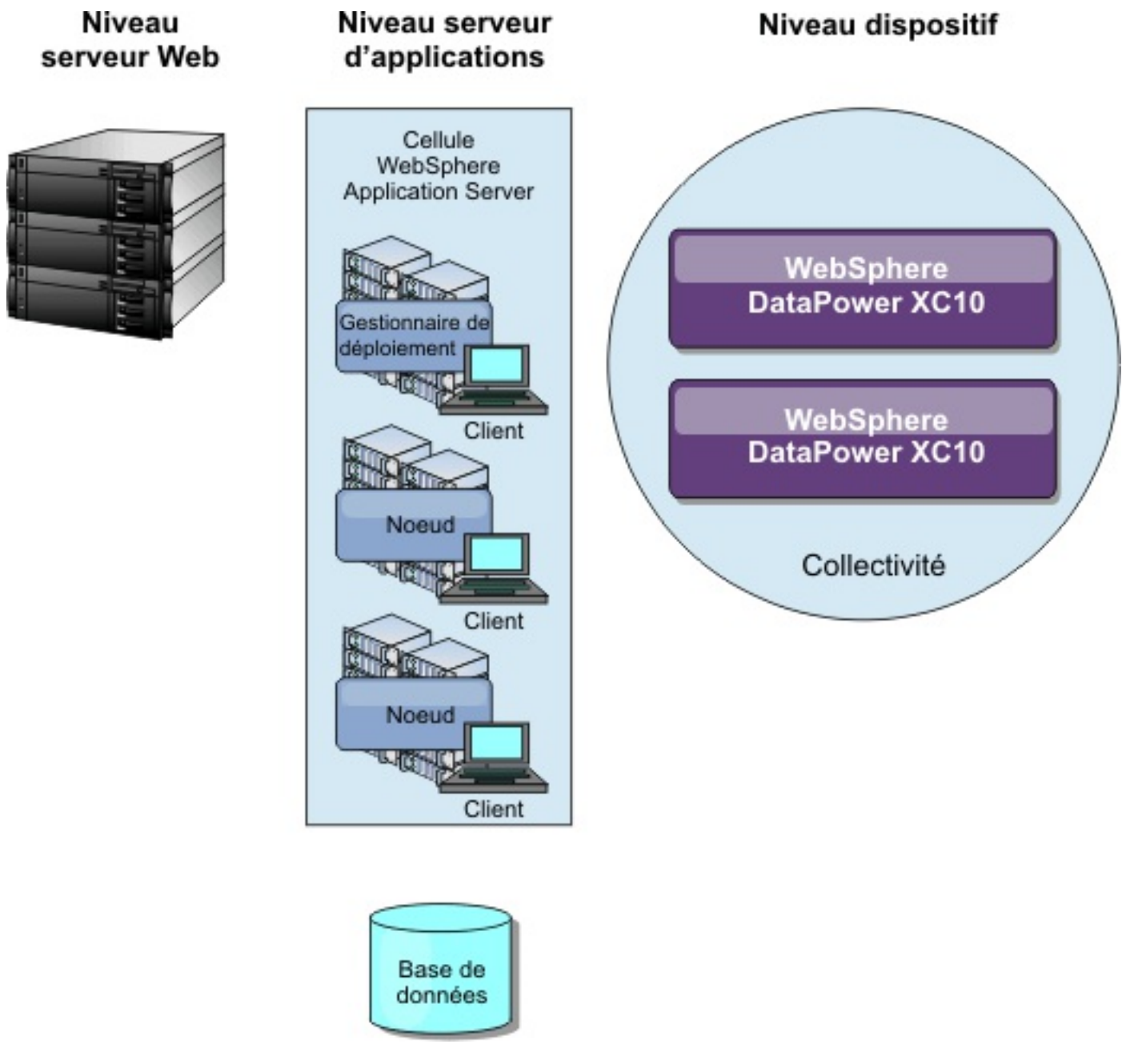
Gestion des utilisateurs simple et flexible

L'interface utilisateur permet de gérer facilement les utilisateurs et les groupes d'utilisateurs de votre dispositif. Elle permet également de créer, de gérer et de contrôler les grilles de données.

Topologie globale

DataPower XC10 Appliance est configuré derrière le niveau serveur d'applications. Au niveau serveur d'applications, vous installez le WebSphere eXtreme Scale Client sur chaque noeud, y compris au niveau du gestionnaire de déploiement. Ce client active la communication entre le niveau serveur d'applications et le niveau dispositif. Au niveau dispositif, vous pouvez regrouper les dispositifs afin de créer une collectivité.

Figure 1. Topologie globale



Nouveautés de la version 2.5

version 2.5 offre des améliorations du contrôle et de la sécurité et prend en charge les grilles de données d'entreprise, les applications .NET, l'interrogation continue et l'invalidation du cache local.

Notes sur l'édition

Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

Topologie du dispositif : collectivités, zones et grilles de données

Une *grille de données* est une unité de stockage qui peut être créée pour stocker les objets d'une application ou d'un ensemble d'applications spécifique. Une *collectivité* les dispositifs afin d'en faciliter l'évolutivité et la gestion. Une *zone* définit un emplacement physique pour votre dispositif et permet de déterminer où les données de la mémoire cache doivent être placées.

2.5+ Présentation de la grille de données d'entreprise

Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Avec le nouveau transport et le nouveau format de sérialisation, vous pouvez connecter des clients Java™ et .NET à une même grille de données.

Présentation du traitement des transactions

WebSphere eXtreme Scale Client utilise des transactions comme mécanisme d'interaction avec les données.

Remarques

Remarques sur les règles de confidentialité

Nouveautés de la version 2.5

version 2.5 offre des améliorations du contrôle et de la sécurité et prend en charge les grilles de données d'entreprise, les applications .NET, l'interrogation continue et l'invalidation du cache local.

WebSphere eXtreme Scale Client for .NET
Installer WebSphere eXtreme Scale Client for .NET vous permet de déployer des applications .NET qui accèdent à la grille de données. Pour en savoir plus...

Etat de démarrage du dispositif
Vous pouvez désormais visualiser l'état de démarrage du dispositif dans l'interface utilisateur. Pour en savoir plus...

Fournisseur de stockage d'état de session ASP.NET
Vous pouvez configurer vos applications ASP.NET pour stocker l'état de session dans la grille de données. Pour en savoir plus...

Interrogation continue
Lorsque vous développez des applications client qui interagissent avec la grille de données, vous pouvez avoir besoin de requêtes qui extraient automatiquement des résultats en temps réel lorsque de nouvelles entrées sont insérées ou mises à jour. Vous pouvez utiliser l'interrogation continue pour être notifié dans votre machine JVM (Java™ virtual machine) lorsque des données sont insérées ou mises à jour dans la grille de données. Cette fonction facilite la gestion des données et de la grille pour les développeurs, les administrateurs, etc. Pour en savoir plus...

Suppression de tâches
Vous pouvez supprimer toutes les tâches dans l'interface utilisateur. Pour en savoir plus...

Mise à jour dynamique des propriétés du client pour .NET
Vous pouvez configurer WebSphere eXtreme Scale Client for .NET pour qu'il détecte de façon dynamique les modifications manuelles apportées aux valeurs de propriété dans les fichiers de propriétés des clients. Il n'est pas nécessaire de redémarrer la connexion à la grille de données pour que ces modifications prennent effet. Pour en savoir plus...

Activation de la norme FIPS
Vous pouvez configurer la collectivité de dispositifs pour qu'elle utilise la norme FIPS (Federal Information Processing Standard) 140-2 pour toutes les communications réseau chiffrées. Cette norme garantit une protection élevée des données transmises. Pour en savoir plus...

Codage de la propriété credentialGeneratorProps dans les fichiers de propriétés de client pour .NET
Vous pouvez coder la valeur de la propriété credentialGeneratorProps dans le fichier `Client.Net.properties` à l'aide de l'utilitaire **`FilePasswordEncoder`**. Pour en savoir plus...

Grille de données d'entreprise
Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Vous pouvez ainsi connecter des clients Java et .NET à une même grille de données. Pour en savoir plus...

eXtreme Data Format (XDF)
XDF repose sur le plug-in MapSerializerPlugin et il est désormais la technologie de sérialisation par défaut utilisée lorsque vous exécutez IBM® eXtremeIO (XIO).

Exemple d'initiation
Vous pouvez utiliser l'exemple d'initiation pour exécuter des applications client Java et .NET qui accèdent aux grilles de données de votre dispositif. Pour en savoir plus...

Alias de grille
Vous pouvez utiliser la passerelle REST pour créer et gérer un alias de grille. Les alias de grille sont utiles lorsque vous devez remplir plusieurs grilles simultanément. Pour en savoir plus...

Gestion des noms d'hôte
La propriété **`publishHost`** vous permet de configurer et de remplacer le nom d'hôte publié. Vous disposez ainsi d'un contrôle plus fin sur la communication entre les clients et les serveurs, ce qui vous permet d'optimiser la communication entre les noeuds dans la grille de données. Pour en savoir plus...

LDAP sur SSL
Sécurisez votre annuaire LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs qui

accèdent à votre dispositif. Pour ce faire, configurez le dispositif pour l'authentification LDAP via une connexion SSL. Pour en savoir plus...

Invalidation du cache local

Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée sur la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Pour en savoir plus...

Nouvelles commandes et nouveaux paramètres de l'utilitaire `xscmd`

- Commande **`xscmd -c getNotificationFilter`** : exécutez cette commande pour afficher les filtres en cours des nouvelles notifications depuis Message Center. Pour en savoir plus...
- Commande **`xscmd -c listenForNotifications`** : exécutez cette commande pour écouter les nouvelles notifications depuis Message Center. Pour en savoir plus...
- Commande **`xscmd -c setNotificationFilter`** : exécutez cette commande pour créer un filtre pour les nouvelles notifications depuis Message Center. Pour en savoir plus...
- Commande **`xscmd -c showLinkedDomains`** : exécutez cette commande pour identifier les domaines de service de catalogue liés au domaine de service de catalogue local. Pour en savoir plus...
- Commande **`xscmd -c showNotificationHistory`** : exécutez cette commande pour afficher la sortie de l'historique des notifications d'événements dans un tableau.
- Commande **`xscmd -c showSessionSize`** : exécutez cette commande pour afficher la taille d'une session. Pour en savoir plus...
- Paramètre **`-to`** ou **`--timeout`** : définissez ce paramètre pour réduire le délai d'attente afin d'éviter d'attendre pendant toute la durée des délais d'attente du système d'exploitation ou du réseau au cours d'un arrêt réseau ou d'une perte du système. Pour en savoir plus...
- Paramètre **`-hc`** ou **`--linkHealthCheck`** : utilisez ce paramètre avec la commande **`xscmd -c showLinkedPrimaries`** pour vérifier si les fragments primaires disposent du nombre approprié de liens de collectivité.
- Commande **`xscmd -c listDisabledForPlacement`** : exécutez cette commande pour afficher la liste des conteneurs de fragments désactivés pour le placement des fragments.
- Commande **`xscmd -c listIndoubts`** : exécutez cette commande pour afficher la liste des transactions en attente de validation. Exécutez cette commande pour résoudre les exceptions potentielles de délai d'attente de verrouillage sur une partition. Pour en savoir plus...
- Commandes **`xscmd -c showReplicationState`** et **`xscmd -c showDomainReplicationState`** : exécutez ces commandes pour afficher l'état des révisions précédentes sur les serveurs de catalogue ou les domaines de service de catalogue. Pour en savoir plus...
- Commande **`xscmd -c showTransport`** : exécutez cette commande pour afficher le type de transport du domaine de service de catalogue. Pour en savoir plus...

Remplacement de la propriété `requestRetryTimeout` dans les grilles de données de cache dynamique

Utilisez la propriété de mémoire cache dynamique **`com.ibm.websphere.xs.dynacache.request_retry_timeout_override`** pour remplacer la propriété **`requestRetryTimeout`** du fichier de propriétés du client et spécifier pendant combien de temps (en millisecondes) une requête peut s'exécuter avant d'expirer. Pour en savoir plus...

Exécution de l'utilitaire `xscmd` à partir de l'interface de ligne de commande du dispositif

Vous pouvez exécuter l'utilitaire **`xscmd`** à partir de l'interface de ligne de commande du dispositif. Dans ce cas, vous n'avez pas besoin de fournir les arguments de ligne de commande pour la sécurité et les connexions de serveur de catalogue. Pour en savoir plus...

Génération de clé secrète

Remplacement de la clé secrète d'authentification définie par défaut en usine par une clé secrète unique Pour en savoir plus...

Mise à jour des statistiques SNMP (Simple Network Monitoring Protocol)

Les bases d'informations de gestion (MIB) fournies avec le WebSphere DataPower XC10 Appliance pour définir les données SNMP disponibles pour le client SNMP ont été mises à jour avec de nouvelles statistiques. Ces dernières incluent applianceUsedBytes, applianceCapacity, gridUsedBytes et gridCapacity. Pour plus d'informations sur l'activation de la surveillance de SNMP et le téléchargement de ce fichier MIB, voir Activation de la surveillance SNMP du dispositif.

Définition du délai d'attente et du délai d'attente pour les requêtes

Vous pouvez spécifier la durée maximum de traitement d'une connexion ou d'une requête, que vous utilisiez le transport eXtremeIO (XIO) ou le transport Object Request Broker (ORB). Par exemple, dans le cas de XIO, vous pouvez utiliser la propriété **`xiotimeout`** pour spécifier le délai d'attente pour les tentatives d'établissement d'une connexion socket sortante, et la propriété **`xioRequestTimeout`** pour spécifier le

nombre de secondes pendant lequel une requête peut attendre avant d'expirer. La propriété **`requestRetryTimeout`** s'applique au transport XIO ou ORB. Elle vous permet de spécifier pendant combien de temps le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Pour en savoir plus...

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Notes sur l'édition

Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

- Dernières mises à jour, limitations et problèmes connus
- Accès à la configuration système et logicielle requise
- Accès à la documentation du produit
- Accès au site de support technique du produit
- Contacter le service de support logiciel IBM

Dernières mises à jour, limitations et problèmes connus

Les notes sur l'édition sont disponibles sur le site de support technique du produit sous forme de notes techniques. Pour afficher une liste de toutes les notes techniques de WebSphere DataPower XC10 Appliance, accédez à la page Web du support technique. Cliquer sur les liens indiqués ici générera une recherche dans la page Web du support des notes sur l'édition correspondantes, lesquelles seront retournées sous forme de liste.

- Pour afficher une liste des notes sur l'édition, accédez à la page Web du support technique.

Accès à la configuration système et logicielle requise

La configuration matérielle et logicielle requise est détaillée dans les pages suivantes :

- Configuration requise

Accès à la documentation du produit

Pour accéder à l'ensemble des informations, accédez à la page Bibliothèque.

Accès au site de support technique du produit

Pour rechercher les informations de support technique et notamment les dernières notes techniques, les fichiers à télécharger et les correctifs, accédez à la page du support technique.

Contacter le service de support logiciel IBM

Si un incident survient lors de l'utilisation du produit, essayez tout d'abord d'effectuer les opérations suivantes :

- Suivez les étapes décrites dans la documentation du produit
- Recherchez la documentation connexe dans l'aide en ligne
- Recherchez les messages d'erreur dans le document de référence des messages

Si vous ne parvenez pas à résoudre l'erreur à l'aide d'une des méthodes précédentes, prenez contact avec le support technique IBM®.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance
Rubrique parent : Traitement des incidents

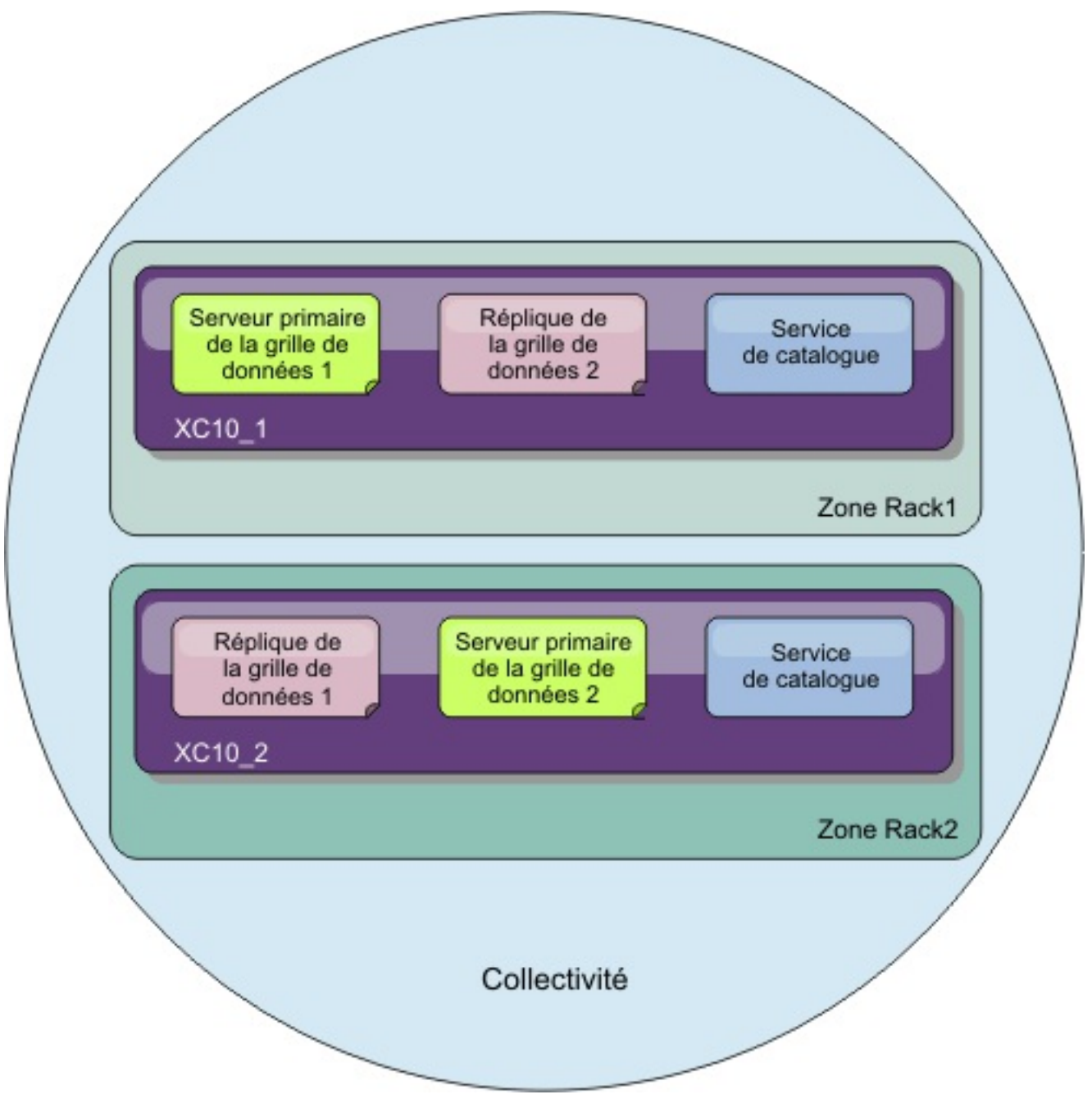
Topologie du dispositif : collectivités, zones et grilles de données

Une *grille de données* est une unité de stockage qui peut être créée pour stocker les objets d'une application ou d'un ensemble d'applications spécifique. Une *collectivité* les dispositifs afin d'en faciliter l'évolutivité et la gestion. Une *zone* définit un emplacement physique pour votre dispositif et permet de déterminer où les données de la mémoire cache doivent être placées.

Topologie du dispositif

Les collectivités et les zones sont associées à une ou plusieurs instances de WebSphere DataPower XC10 Appliance. Chaque dispositif peut être membre d'une collectivité et d'une zone. Chaque dispositif héberge plusieurs grilles de données, qui contiennent les données en mémoire cache.

Figure 1. Topologie des collectivités et des zones



Important : Deux dispositifs sont nécessaires pour rendre la grille de données hautement disponible.

Collectivités et réplication multimaître

La réplication multimaître est une technique qui permet de garantir une disponibilité continue dans plusieurs environnements de déploiement. Des topologies multimaîtres peuvent être mises en oeuvre dans WebSphere DataPower XC10 Appliance en plusieurs collectivités et en les associant. Lorsque vous définissez une collectivité, les informations suivantes sont partagées entre les dispositifs de la collectivité : grilles de données, informations de contrôle, membres de la collectivité et de zone et utilisateurs. Lorsque vous mettez à jour ces informations, les modifications apportées affectent tous les autres dispositifs de la collectivité. Le *service de catalogue* permet la communication entre les dispositifs. Le service de catalogue est un groupe de serveurs de catalogue. Les différents dispositifs de la collectivité exécutent un serveur de catalogue, dans la limite de trois serveurs de catalogue par collectivité. Si vous disposez de plus de trois dispositifs au sein d'une collectivité, le service de catalogue s'exécute sur les trois premiers dispositifs ajoutés à la collectivité. Si vous supprimez un dispositif associé à un serveur de catalogue de la collectivité, ou si un serveur de catalogue devient disponible, le dispositif suivant ajouté à la collectivité exécute le serveur de catalogue. Ce dernier ne bascule pas vers les autres dispositifs.

Pour ajouter un dispositif à une collectivité, ajoutez le nom d'hôte et la clé secrète du dispositif au panneau de configuration de la collectivité à partir d'un autre dispositif. Cette configuration peut s'effectuer à partir de tous les dispositifs de la collectivité. En effet, l'appartenance à la collectivité est conservée pour tous les membres de la collectivité.

Les dispositifs peuvent appartenir à une seule collectivité. Vous ne pouvez pas ajouter un dispositif appartenant déjà à une collectivité à une autre collectivité. Il est également impossible de regrouper deux

collectivités dans une collectivité unique. Pour joindre les dispositifs de différentes collectivités, vous devez supprimer ces dispositifs de leur collectivité respective et les définir ainsi en tant que dispositifs autonomes. Vous pouvez ensuite créer une collectivité regroupant l'ensemble de ces dispositifs.

Vous pouvez utiliser une collectivité pour effectuer la plupart des modifications de configuration. Vous devez toutefois vous connecter à un dispositif donné pour modifier les paramètres des panneaux **Dispositifs** > **Paramètres de dispositif** et **Dispositifs** > **Identification et résolution des incidents**.

Zones

Les zones sont associées à l'emplacement physique du dispositif (par exemple, une ville ou un emplacement d'armoire au sein d'un lab). Les zones permettent au service de catalogue de définir l'emplacement de stockage des données dans vos grilles de données. Par exemple, si les données principales de la grille de données sont stockées dans une zone donnée, les données secondaires sont stockées sur un dispositif situé dans une zone différente. Dans cette configuration, un basculement peut se produire entre un système principal et une réplique si le dispositif contenant les données principales de grille de données tombe en panne.

Grilles de données

Grilles de données regroupe les objets correspondant aux applications. En plaçant des objets en mémoire cache, vous pouvez augmenter les performances de vos applications. Il existe trois types de grilles de données :

grille de données simple

Les grilles de données simples regroupent les données sous forme de paires clé-valeur. Par exemple, vous pouvez stocker les résultats d'une requête de base de données dans une grille de données simple. L'implémentation d'une grille de données simple s'effectue à l'aide de l'API ObjectMap. Le fonctionnement de l'API ObjectMap est similaire à celui des mappes Java™.

grille de données de session

Si vous utilisez des sessions WebSphere Application Server, vous pouvez configurer votre application de manière à utiliser une grille de données de session sur le dispositif des données de gestion de session. Vous pouvez configurer votre application de manière à utiliser une grille de données de session lorsque vous installez une nouvelle application. Vous pouvez également mettre à jour les paramètres de vos applications ou de vos serveurs existants de manière à utiliser une grille de données de session sur le dispositif.

grille de données de mémoire cache dynamique

Une grille de données de mémoire cache dynamique définie sur un dispositif permet de stocker des données en provenance de la mémoire cache dynamique de WebSphere Application Server. Vous pouvez activer des applications rédigées à l'aide de l'API de cache dynamique ou les applications utilisant la mise en cache au niveau du conteneur ,par exemple, les servlets, pour l'utilisation du dispositif en tant que fournisseur de mémoire cache. Les serveurs d'applications utilisent dès lors une quantité de mémoire inférieure. Toutes les données de mémoire cache sont transférées vers le dispositif et disparaisse de la mémoire des serveurs d'applications.

Répliques de grilles de données

Vous pouvez définir un nombre cible de répliques pour une grille de données spécifique. Des répliques sont créées lorsque la collectivité contient au moins deux dispositifs. Si elle ne contient qu'un seul dispositif, aucune réplique n'est créée. Si vous avez un nombre de dispositifs *n* dans votre collectivité, le nombre maximal de répliques est *n-1* car un des dispositifs héberge la grille de données principale. Si votre nombre cible de répliques est supérieur à la valeur *n-1* actuelle, davantage de répliques peuvent être placées lorsque vous ajoutez des dispositifs à la collectivité. Envisagez de définir le nombre de répliques au nombre de répliques le plus élevé que vous pourrez souhaiter dans le futur. La modification des paramètres des répliques requiert l'effacement des grilles de données : définissez donc la valeur en tenant compte du nombre futur de répliques. Quand de nouveaux dispositifs rejoignent la collectivité, des répliques supplémentaires sont créées. Les grilles de données principales et répliques sont réparties de façon homogène, ou segmentées, sur tous les dispositifs de la collectivité. Quand de nouveaux dispositifs rejoignent la collectivité, un nouvel équilibrage a lieu pour répartir les grilles de données principales et répliques.

Les répliques peuvent être synchrones ou asynchrones. Les répliques synchrones reçoivent des mises à jour dans le cadre de la transaction sur la grille de données principale. Les répliques asynchrones sont mises à jour après la validation de la transaction sur la grille de données principale. Les répliques synchrones garantissent la cohérence des données. En contrepartie, elles augmentent le délai de traitement des requêtes par rapport aux répliques asynchrones. Les répliques asynchrones n'offrent pas les mêmes garanties en termes de cohérence des données, mais accélèrent les délais de traitement des transactions. Une grille de données possède une réplique asynchrone par défaut. Un algorithme de placement contrôle l'emplacement des répliques.

Mappes

Les mappes sont les structures de données qui contiennent les données de la grille de données en paires clé-valeur. Une grille de données unique peut avoir plusieurs mappes résidant dans les grilles de données et les répliques de grille de données.

Vous pouvez créer des mappes supplémentaires dans la grille de données en demandant à votre application client de se connecter à une mappe spécifiquement nommée. Une mappe dynamique est automatiquement créée.

Liaisons de collectivité

Une seule collectivité ne doit pas couvrir un réseau non fiable car des incidents de faux positif risquent d'être détectés. Cependant, vous pouvez être amené à répliquer des données de grille de données sur les dispositifs dont la connectivité du réseau n'est pas fiable. Voici quelques scénarios courants dans lesquels vous pouvez être amené à utiliser ce type de topologie :

- Reprise après incident entre des centres de données dans lesquels une collectivité est active et une autre est utilisée à des fins de secours
- Centres de données géographiquement répartis dans lesquels toutes les collectivités sont actives pour les clients géographiquement proches

Une fois que vous connectez deux collectivités, toutes les grilles de données portant le même nom sont répliquées de façon asynchrone d'une collectivité à l'autre. Ces grilles de données doivent comporter le même nombre de répliques dans chaque collectivité et posséder les mêmes configurations de mappe dynamique.

> **Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître**
> Vous disposez de plusieurs options pour choisir la topologie de votre déploiement qui intègre plusieurs collectivités. Des topologies multimaîtres peuvent être mises en oeuvre dans DataPower XC10 Appliance en plusieurs collectivités et en les associant.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

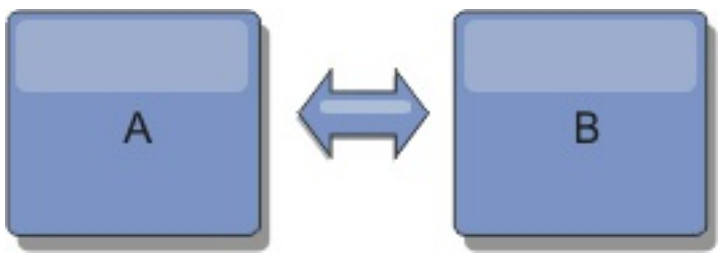
Tâches associées:
Configuration d'une réplication multimaître entre les collectivités

Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître

Vous disposez de plusieurs options pour choisir la topologie de votre déploiement qui intègre plusieurs collectivités. Des topologies multimaîtres peuvent être mises en oeuvre dans DataPower XC10 Appliance en plusieurs collectivités et en les associant.

Liaisons connectant les collectivités

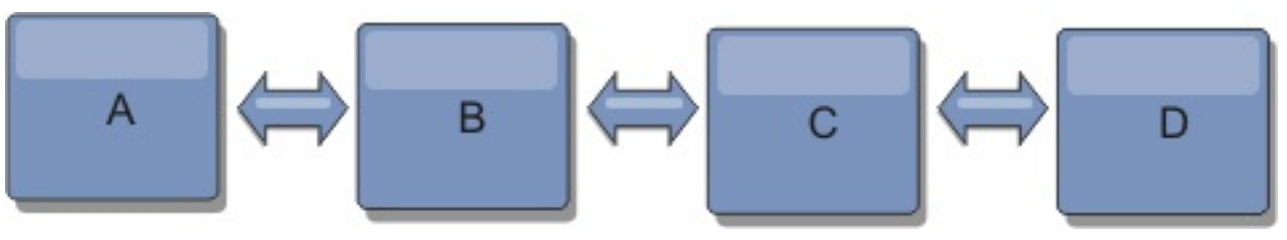
Une infrastructure de grilles de données de réplication est un graphique de collectivités interconnectés avec des liaisons bidirectionnelles. Avec une liaison, deux collectivités peuvent communiquer les modifications de données. Par exemple, la topologie la plus simple est une paire de collectivités avec une liaison unique entre eux. Les collectivités sont nommés par ordre alphabétique: A, B, C, etc., à partir de la gauche. Une liaison peut traverser un réseau WAN (wide area network) pour couvrir une grande distance. Même si la liaison est interrompue, vous pouvez toujours modifier les données dans une collectivité. La topologie rapproche les modifications quand la liaison reconnecte les collectivités. Les liaisons tentent automatiquement de se reconnecter si la connexion réseau est interrompue.



Après avoir établi les liaisons, le produit tente d'abord de rendre chaque collectivité identique. Ensuite, eXtreme Scale tente de maintenir identiques les conditions à mesure que des modifications se produisent dans un collectivité. L'objectif vise à faire de chaque collectivité le miroir exact d'un autre collectivité connecté par les liaisons. Les liaisons de réplication entre les collectivités permettent de copier une modification effectuée dans un collectivité vers les autres collectivités.

Topologies linéaires

Même s'il s'agit d'un déploiement simple, une topologie linéaire montre certaines qualités des liaisons. Tout d'abord, il n'est pas nécessaire qu'un collectivité soit directement connecté à chacun des autres domaines de service de catalogue pour recevoir des modifications. Le collectivité B extrait les modifications du collectivité A. Le collectivité C reçoit les modifications du collectivité A via le collectivité B, lequel connecte les domaines A et C. De même, le collectivité D reçoit les modifications des autres collectivités via le collectivité C. Cette fonction répartit la charge de distribution des modifications en l'éloignant de la source des modifications.



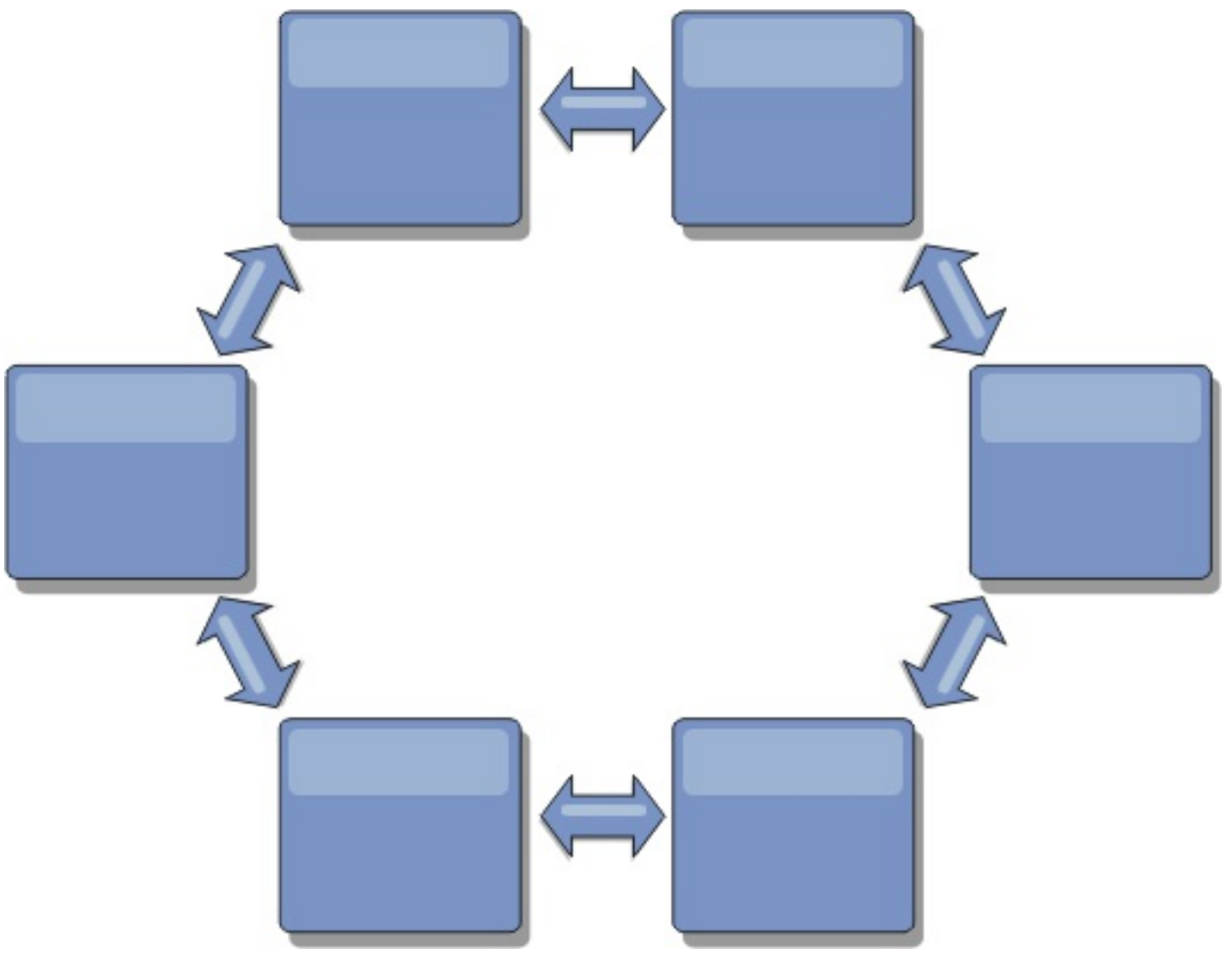
Notez que si le collectivité C est défaillant, les actions suivantes se produisent :

1. Le collectivité D serait orphelin jusqu'au redémarrage du collectivité C.
2. Le collectivité C doit se synchroniser avec le collectivité B, lequel est une copie du collectivité A.
3. Le collectivité D utilise le collectivité C pour se synchroniser avec les modifications des domaines de service de catalogues A et B. Ces modifications se sont produites initialement lorsque le collectivité D étaient orphelin (lorsque le collectivité C était arrêté).

Enfin, les collectivités A, B, C et D, sont de nouveau tous identiques.

Topologies en anneau

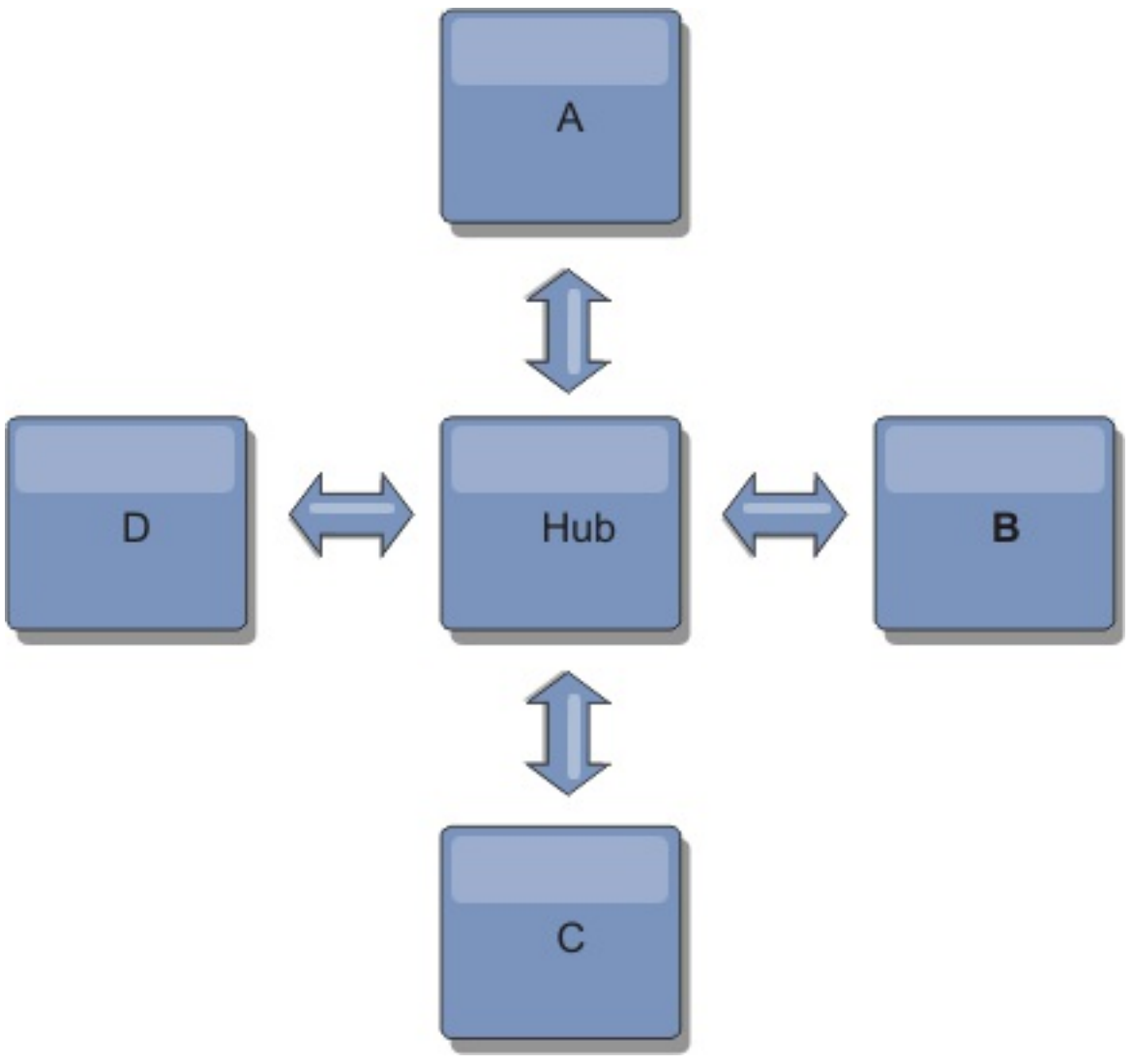
Les topologies en anneau sont un exemple de topologie encore plus résilientes. Lorsqu'un collectivité ou une liaison unique est défaillant, les collectivités restants peuvent encore obtenir des modifications. Les collectivités parcourent l'anneau en s'éloignant de la défaillance. Chaque collectivité possède au maximum deux liens vers d'autres collectivités, quelle que soit la taille de la topologie en anneau. Le délai de propagation des modifications peut être important. Les modifications d'un collectivité particulier peuvent devoir traverser plusieurs liaisons pour que tous les collectivités aient les modifications. Une topologie linéaire a la même caractéristique.



Vous pouvez également déployer une topologie en anneau plus sophistiquée, avec un collectivité racine au centre de l'anneau. Le collectivité racine fait office de point central de rapprochement. Les autres collectivités font office de points distants de rapprochement pour les modifications se produisant dans le collectivité racine. Le collectivité racine peut arbitrer les modifications entre les collectivités. Si une topologie en anneau contient plusieurs anneaux autour d'un collectivité racine, le collectivité ne peut pas arbitrer les modifications dans la partie interne de l'anneau. Toutefois, les résultats de l'arbitrage sont propagés dans les collectivités des autres anneaux.

Topologies en étoile

Avec une topologie en étoile, les modifications parcourent un collectivité du concentrateur. Etant donné que le concentrateur est le seul collectivité intermédiaire spécifié, les topologies en étoile ont une latence inférieure. Le collectivité du concentrateur est connecté à chaque branche de collectivité via une liaison. Le concentrateur répartit les modifications entre les collectivités. Il fait office de point de rapprochement pour les collisions. Dans un environnement soumis à une fréquence élevée de modifications, le concentrateur peut avoir besoin de s'exécuter sur plus de matériels que les branches pour rester synchronisé. WebSphere DataPower XC10 Appliance est conçu pour évoluer de manière linéaire, ce qui signifie que l'on peut, si nécessaire, étoffer le concentrateur sans difficultés. Toutefois, si le concentrateur tombe en panne, les modifications ne sont pas distribuées jusqu'à ce qu'il redémarre. Toutes les modifications sur les branches du sous-domaine de service de catalogue seront réparties après la reconnexion du concentrateur.



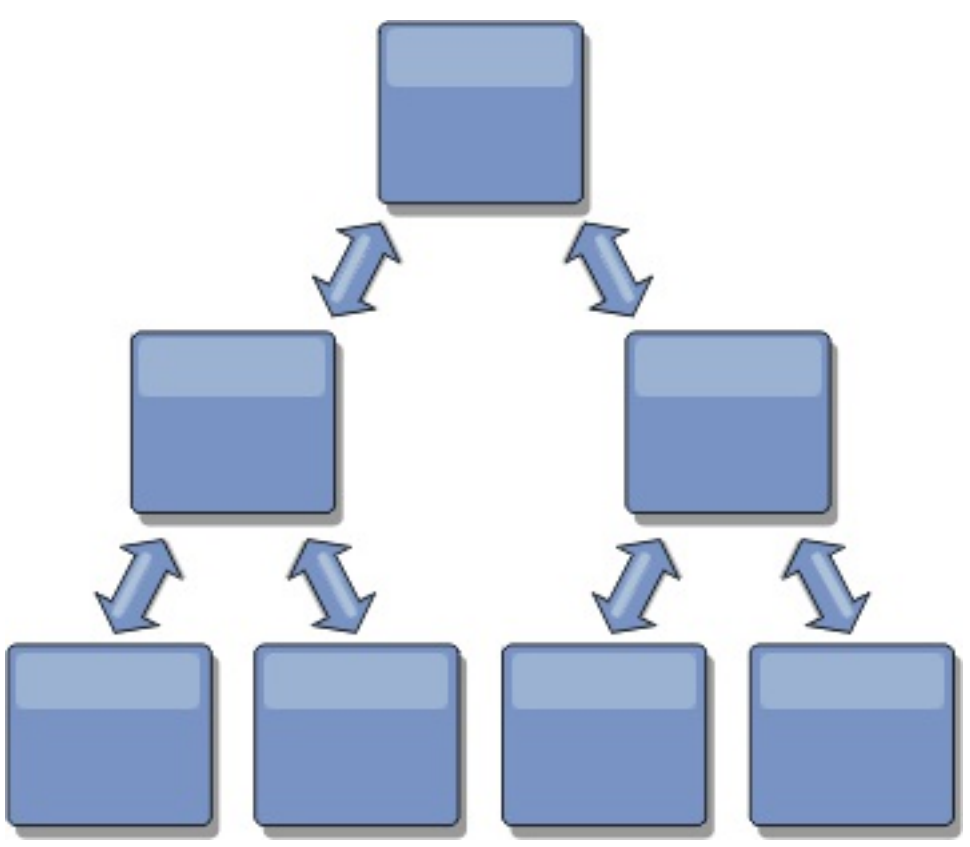
Vous pouvez également utiliser une stratégie avec les clients intégralement répliqués, une variante de la topologie qui utilise une paire de serveurs s'exécutant comme concentrateur. Chaque client crée une grille de données à conteneur unique autonome avec un catalogue dans la machine virtuelle Java client. Un client utilise sa grille de données pour se connecter au catalogue du concentrateur. Cette connexion provoque la synchronisation du client avec le concentrateur dès que le client obtient une connexion au concentrateur.

Toutes les modifications effectuées par le client sont locales pour le client et elles sont répliquées vers le concentrateur de manière asynchrone. Le concentrateur joue le rôle de collectivité d'arbitrage, répartissant les modifications à tous les clients connectés. La topologie de clients intégralement répliqués fournit un cache L2 fiable pour un associateur relationnel d'objets comme OpenJPA. Les modifications sont réparties rapidement via le concentrateur entre les machines virtuelles client. Si la taille du cache peut être contenue dans le segment de mémoire disponible, la topologie est une architecture fiable pour ce style de cache L2.

Si nécessaire, utilisez plusieurs partitions pour échelonner le collectivité concentrateur sur plusieurs machines virtuelles Java. Etant donné que toutes les données doivent toujours tenir sur une seule machine virtuelle Java client, plusieurs partitions augmentent la capacité du concentrateur à répartir et à arbitrer les modifications. Cependant, plusieurs partitions ne changent pas la capacité d'un collectivité unique.

Topologies en arbre

Vous pouvez également utiliser un arbre dirigé acyclique. Un arbre acyclique n'a pas de cycles ou de boucles, et une configuration dirigée limite les liaisons aux parents et enfants existants uniquement. Cette configuration est utile pour les topologies disposant d'un grand nombre de collectivités. Dans ces topologies, il n'est pas pratique d'avoir un concentrateur central connecté à chaque branche. Ce type de topologie peut également être utile lorsque vous devez ajouter des collectivités enfant sans mettre à jour le collectivité racine.



Une topologie en arbre peut toujours avoir un point central de rapprochement dans le collectivité racine. Le deuxième niveau peut toujours fonctionner en tant que point de rapprochement distant pour les modifications se produisant dans le collectivité en dessous. Le collectivité racine peut arbitrer les modifications entre les collectivités sur le deuxième niveau uniquement. Vous pouvez également utiliser des arbres n-aire ayant chacun n enfants à chaque niveau. Chaque collectivité se connecte à *n* liaisons.

Clients intégralement répliqués

Cette variante de la topologie implique une paire de serveurs s'exécutant comme concentrateur. Chaque client crée une grille de données à conteneur unique autonome avec un catalogue dans la machine virtuelle Java client. Un client utilise sa grille de données pour se connecter au catalogue du concentrateur, ce qui provoque la synchronisation du client avec le concentrateur dès que le client obtient une connexion au concentrateur.

Toutes les modifications effectuées par le client sont locales pour le client et elles sont répliquées vers le concentrateur de manière asynchrone. Le concentrateur joue le rôle de collectivité d'arbitrage, répartissant les modifications à tous les clients connectés. La topologie de clients intégralement répliqués fournit un bon cache de niveau 2 pour un associateur relationnel d'objets comme OpenJPA. Les modifications sont réparties rapidement via le concentrateur entre les machines virtuelles client. Tant que la taille du cache peut être contenue dans l'espace de segment mémoire disponible des clients, cette topologie est une architecture tout à fait indiquée pour ce style de cache de niveau 2.

Si nécessaire, utilisez plusieurs partitions pour échelonner le collectivité concentrateur sur plusieurs machines virtuelles Java. Toutes les données devant tenir sur une seule machine virtuelle Java, l'utilisation de partitions multiples augmente la capacité du concentrateur à répartir et à arbitrer les modifications, mais elle ne change pas la capacité d'un collectivité unique.

Rubrique parent : Topologie du dispositif : collectivités, zones et grilles de données

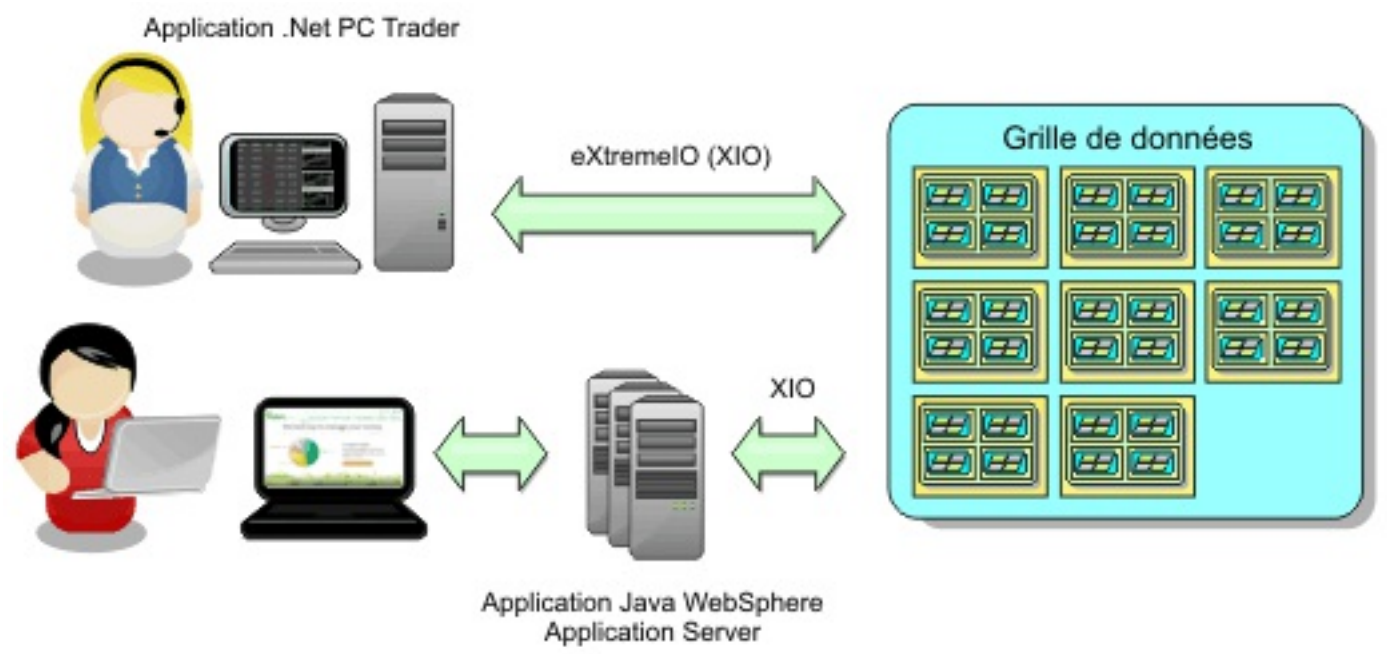
Tâches associées:
Configuration d'une réplication multimaître entre les collectivités

Présentation de la grille de données d'entreprise

Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Avec le nouveau transport et le nouveau format de sérialisation, vous pouvez connecter des clients Java™ et .NET à une même grille de données.

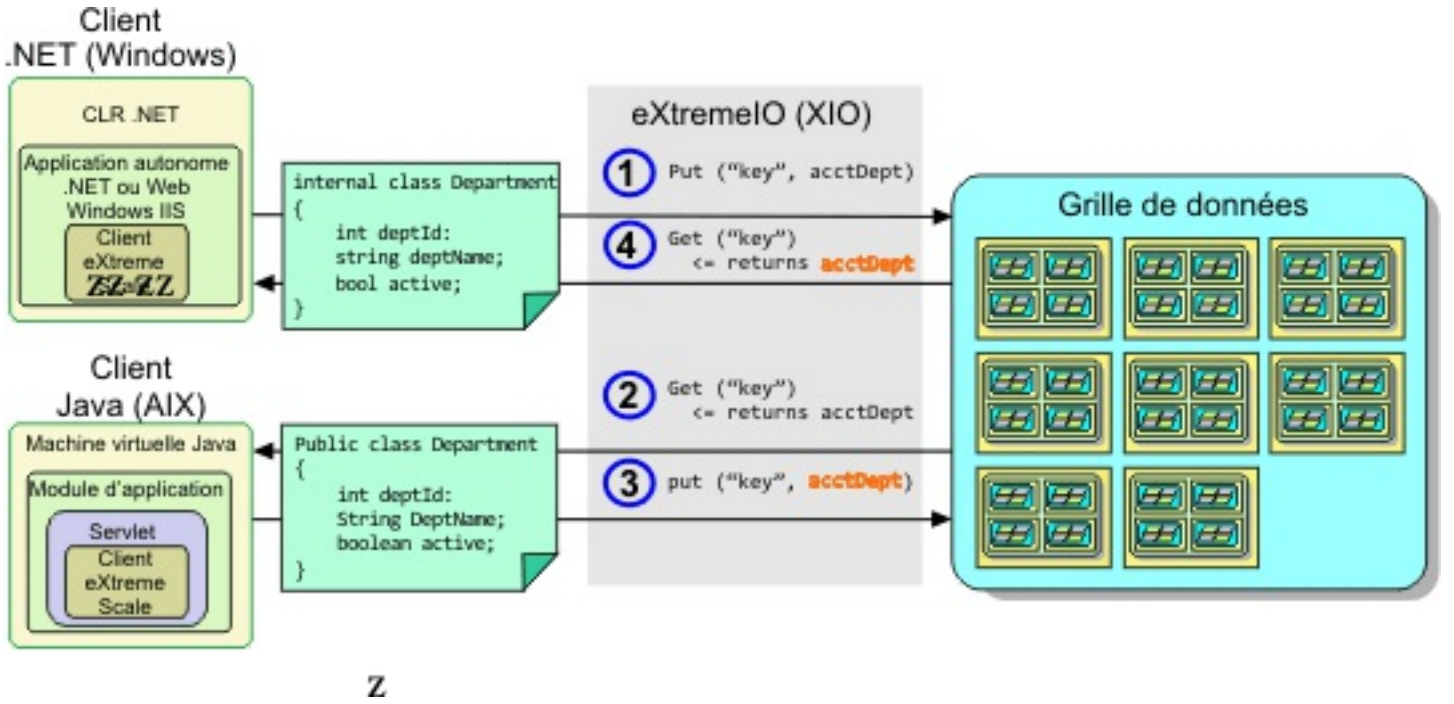
Avec la grille de données d'entreprise, vous pouvez créer plusieurs types d'applications, écrites dans divers langages de programmation, pour accéder aux mêmes objets dans la grille de données. Dans les versions antérieures, les applications de grille de données devaient être écrites en Java uniquement. Avec la fonction de grille d'entreprise, vous pouvez écrire des applications .NET qui créent, extraient, mettent à jour et suppriment des objets de la même grille de données que l'application Java.

Figure 1. Présentation générale de la grille de données d'entreprise



Mises à jour d'objets dans différentes applications

Figure 2. Flux de mise à jour d'un objet dans la grille de données d'entreprise



Z

1. Le client .NET enregistre les données dans son format dans la grille de données.
2. Les données sont stockées dans un format universel pour que lorsque le client Java les demande, elles puissent être converties dans le format Java.
3. Le client Java met à jour et enregistre de nouveau les données.
4. Le client .NET accède aux données mises à jour, période au cours de laquelle les données sont converties dans le format .NET.

Mécanisme de transport

eXtremeIO (XIO) est un protocole de transport multiprotocole. XIO remplace le courtier ORB (Object Request Broker) Java. Avec le courtier ORB, WebSphere DataPower XC10 Appliance est lié aux applications client natives Java. XIO est un mécanisme de transport personnalisé dédié à la mise en cache et qui permet aux applications client dans différents langage de programmation de se connecter à la grille de données.

Format de sérialisation

Le format de données XDF (eXtreme data format) est un format de sérialisation multiplateforme XDF remplace la sérialisation Java dans les mappes dont l'attribut CopyMode a la valeur `C0PY_T0_BYTES` dans le fichier XML descripteur ObjectGrid. XSF améliore les performances et rend les données plus compactes. En outre, XDF permet aux applications client dans différents langages de programmation de se connecter à la

même grille de données.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance
Rubrique parent : Scénario : Configuration d'une grille de données d'entreprise
Rubrique suivante : Configuration d'IBM eXtremeIO (XIO)

Présentation du traitement des transactions

WebSphere eXtreme Scale Client utilise des transactions comme mécanisme d'interaction avec les données.

Java

Traitement des transactions dans les applications Java™

Pour pouvoir interagir avec les données, l'unité d'exécution de votre application a besoin de sa propre session. Lorsque l'application souhaite utiliser ObjectGrid sur une unité d'exécution, appelez l'une des méthodes ObjectGrid.getSession pour obtenir une session. Avec cette session, l'application peut travailler avec les données stockées dans les mappes ObjectGrid.

Lorsqu'une application utilise un objet Session, la session doit être dans le contexte d'une transaction. Une transaction commence et se valide ou commence et s'annule en utilisant les méthodes begin, commit et rollback sur l'objet Session. Les applications peuvent également utiliser le mode de validation automatique, dans lequel la Session initie et valide automatiquement une transaction lorsqu'une opération est effectuée sur la mappe. Ce mode d'auto-validation ne permet pas de regrouper des opérations multiples en une seule transaction. Il s'agit donc de l'option la plus lente si vous créez un lot d'opérations multiples dans une seule transaction. Cependant, pour les transactions contenant une seule opération, l'auto-validation est l'option la plus rapide.

Lorsque l'application n'utilise plus la session, utilisez la méthode facultative Session.close() pour fermer la session. La fermeture de la session a pour effet de libérer cette dernière du segment de mémoire et de permettre de réutiliser des appels ultérieurs vers la méthode getSession(), ce qui améliore les performances.

.NET

Traitement des transactions dans les applications .NET

Pour pouvoir interagir avec les données, chaque unité d'exécution de votre application a besoin de son propre objet de transaction. Pour utiliser l'interface IGrid sur une unité d'exécution de votre application, appelez l'une des méthodes suivantes :

- IGrid.GetGridMapPessimisticAutoTx
- IGrid.GetGridMapPessimisticTx

Lorsque vous appelez ces méthodes, vous obtenez un objet IGridMap qui possède un objet de transaction unique. Avec cet objet IGridMap, l'application peut interagir avec les données stockées dans les mappes IGrid. Lorsqu'une application utilise un objet IGridMapPessimisticTx, les opérations effectuées sur la grille de données doivent se situer dans le contexte d'une transaction. Une transaction commence et se valide ou commence et s'annule en utilisant les méthodes begin, commit et rollback sur l'objet IGridTransaction. Les applications peuvent également utiliser le mode de validation automatique, dans lequel l'IGridMapPessimisticAutoTx initie et valide automatiquement une transaction lorsqu'une opération est effectuée sur la mappe. Ce mode d'auto-validation ne permet pas de regrouper des opérations multiples en une seule transaction. Il s'agit donc de l'option la plus lente si vous créez un lot d'opérations multiples dans une seule transaction. Cependant, pour les transactions contenant une seule opération, l'auto-validation est l'option la plus rapide.

Lorsque votre application n'utilise plus l'instance IGridMap, vous pouvez supprimer l'objet IGridMap. Cette suppression entraîne la fermeture de l'objet de transaction associé. Par conséquent, les appels ultérieurs des méthodes GetGridMapPessimisticAutoTx et GetGridMapPessimisticTx peuvent réutiliser un objet de transaction libre existant, ce qui améliore les performances.

Transactions

Les transactions offrent de nombreux avantages pour le stockage et la manipulation de données. Vous pouvez utiliser des transactions pour protéger la grille de données contre les modifications simultanées, appliquer plusieurs modifications comme unité simultanée, répliquer des données et implémenter un cycle de vie pour les verrous appliqués aux modifications.

Stratégies de verrouillage

Les stratégies de verrouillage disponibles sont les suivantes : pessimiste, optimiste et aucune.

Types de verrou

Lorsque vous utilisez les verrouillages pessimiste et optimiste, des verrous partagés (S), pouvant être mis à niveau (U) et exclusifs (X) sont utilisés pour maintenir la cohérence. Les verrous optimistes n'étant pas maintenus, c'est lorsque vous configurez le verrouillage pessimiste que les effets du verrouillage sont le plus apparents. Les verrous ont des cycles de vie. Les différents types de verrou ne sont pas tous compatibles entre eux de la même manière. Les verrous doivent être traités dans l'ordre approprié pour éviter les situations d'interblocage.

Interblocages
Un interblocage peut se produire lorsque deux transactions tentent de mettre à jour la même entrée de cache.

Accès aux données et transactions
WebSphere eXtreme Scale Client utilise des transactions. Dès lors qu'une application dispose d'une connexion à une grille de données, vous pouvez accéder aux données et interagir avec celles-ci dans la grille.

Isolement des transactions
Vous pouvez utiliser l'un des trois niveaux d'isolement de transaction suivants afin d'optimiser la sémantique de verrouillage qui maintient la cohérence dans chaque mappe de cache : lecture reproductible, lecture validée et lecture non validée.

Java **Validation en deux phases et reprise sur incident**
Le protocole de validation en deux phases coordonne toutes les partitions qui participent à une transaction répartie, en déterminant si la transaction doit être validée ou annulée.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Transactions

Les transactions offrent de nombreux avantages pour le stockage et la manipulation de données. Vous pouvez utiliser des transactions pour protéger la grille de données contre les modifications simultanées, appliquer plusieurs modifications comme unité simultanée, répliquer des données et implémenter un cycle de vie pour les verrous appliqués aux modifications.

Quand une transaction démarre, WebSphere eXtreme Scale Client alloue une mappe spéciale des différences pour contenir les changements en cours ou des copies des paires clé-valeur que la transaction utilise. En règle générale, quand un accès à une paire clé-valeur se produit, la valeur est copiée avant que l'application ne la reçoive. Dans les applications Java™, la mappe des différences assure le suivi de toutes les modifications pour les opérations telles que insert, update, get et remove. Dans les applications .NET, la mappe des différences assure le suivi des modifications pour les opérations add, replace, get et remove. Les clés ne sont pas copiées, car elles sont considérées comme non modifiables. Si une transaction est annulée, les informations de la mappe des différences sont supprimées et les verrous sur les entrées sont libérés. Quand une transaction est validée, les changements sont appliqués à la mappe et les verrous sont libérés.

Java Si un objet ObjectTransformer est spécifié dans une application Java, il est utilisé pour copier la valeur. Si la transaction utilise le verrouillage optimiste, les images précédentes des valeurs sont également suivies afin d'être comparées quand la transaction est validée.

Java Si le verrouillage optimiste est utilisé dans une application Java, WebSphere eXtreme Scale Client compare les versions des images précédentes des valeurs avec les valeurs qui se trouvent dans la mappe. Ces valeurs doivent correspondre pour que la transaction soit validée. Cette comparaison autorise un plan de verrouillage à version multiple, mais au prix de la création de deux copies quand la transaction accède à l'entrée. Toute les valeurs sont à nouveau copiées et la nouvelle copie est stockée dans la mappe. WebSphere eXtreme Scale Client crée cette copie pour se protéger contre le fait que l'application change sa référence à la valeur après validation.

Il est possible de ne pas utiliser plusieurs copies des informations. L'application peut enregistrer une copie en utilisant un verrouillage pessimiste au lieu d'un verrouillage optimiste, au prix d'une limitation des accès simultanés. La copie de la valeur au moment de la validation peut également être évitée si l'application accepte de ne pas changer la valeur après une validation.

.NET **Remarque :** Les applications .NET ne prennent en charge que le verrouillage pessimiste.

Avantages des transactions

Utilisez les transactions pour les raisons suivantes :

En utilisant des transactions, vous pouvez :

- annuler les modifications si une exception se produit ou si la logique application requiert l'annulation des changements d'état,
- appliquer plusieurs changements en tant qu'unité atomique au moment de la validation,
- verrouiller et déverrouiller les données afin d'appliquer des changements multiples en tant qu'unité atomique au moment de la validation,
- protéger une unité d'exécution contre les modifications simultanées,
- implémenter un cycle de vie pour les verrous sur les changements,
- produire une unité de réplication atomique.

Taille des transactions

Les transactions volumineuses sont plus efficaces, en particulier pour la réplication. Cependant, les grandes transactions peuvent avoir un effet néfaste sur les accès simultanés car les verrous sur les entrées sont maintenus plus longtemps. Si vous utilisez de grandes transactions, vous pouvez accroître les performances de réplication. Cette augmentation des performances est importante lors du préchargement d'une mappe. Essayez différentes tailles de lot pour déterminer ce qui fonctionne le mieux pour votre scénario.

Mode de validation automatique

Java Si aucune transaction n'a démarré activement, quand une application interagit avec un objet ObjectMap, une opération automatique de démarrage et de validation est effectuée pour l'application. Cette opération fonctionne mais elle empêche l'annulation et le verrouillage de fonctionner efficacement. La vitesse de réplication synchrone est affectée à cause de la très petite taille des transactions. Si vous utilisez une application de gestion des entités, n'utilisez pas le mode de validation automatique car les objets recherchés avec la méthode EntityManager.find deviennent immédiatement non gérés au retour de la méthode et deviennent inutilisables.

.NET Dans les applications .NET, l'interface de mappe GridMapPessimisticAutoTx fournit les opérations begin et commit automatiques équivalentes. Les limitations sont les mêmes : l'annulation et le verrouillage ne fonctionnent pas correctement et la vitesse de réplication synchrone est réduite.

Java

Intégration des transactions Java EE

WebSphere eXtreme Scale Client comporte un adaptateur de ressources conforme à Java Connector Architecture (JCA) 1.5 qui prend en charge les connexions client à une grille de données distante et la gestion des transactions locales. Les applications Java Platform, Enterprise Edition (Java EE) telles que des servlets, des fichiers JavaServer Pages (JSP) et des composants Enterprise JavaBeans (EJB), peuvent démarquer les transactions WebSphere eXtreme Scale Client à l'aide de l'interface javax.resource.cci.LocalTransaction standard ou de l'interface de session WebSphere eXtreme Scale Client.

Lorsque de l'exécution dans WebSphere Application Server avec le support LPS (Last Participant Support) activé dans l'application, vous pouvez inscrire la transaction WebSphere eXtreme Scale Client dans une transaction globale avec d'autres ressources transactionnelles de validation en deux phases.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Programmation de transactions dans des applications .NET

Stratégies de verrouillage

Les stratégies de verrouillage disponibles sont les suivantes : pessimiste, optimiste et aucune.

Les verrous sont liés aux transactions. Vous pouvez spécifier les paramètres de verrouillage suivants :

Java **Aucun verrouillage**

L'exécution sans verrouillage est la plus rapide. Si vous utilisez des données en lecture seule, vous n'avez peut-être pas besoin de verrouillage.

Restriction : Les mappes de sauvegarde configurées pour utiliser une stratégie de non-verrouillage ne peuvent pas participer à une transaction multipartition.

Java .NET **Verrouillage pessimiste**

Place des verrous sur les entrées, puis les maintient jusqu'au moment de la validation. Cette stratégie offre une bonne cohérence au prix d'une réduction de la rapidité de traitement.

Java **Verrouillage optimiste**

Prend une image avant de chaque enregistrement sur lequel la transaction porte et compare cette image avec les valeurs d'entrées en cours quand la transaction est validée. Si les valeurs d'entrées changent, la transaction est annulée. Aucun verrou n'est maintenu jusqu'au moment de la validation. Cette stratégie de verrouillage offre un meilleur accès simultané que les stratégies pessimistes, au risque que la transaction soit annulée et au prix de la mémoire nécessaire pour une copie supplémentaire de l'entrée.

Important : Si vous utilisez une application client avec WebSphere eXtreme Scale Client for .NET, seul le verrouillage pessimiste est pris en charge.

Gestionnaire de verrous

Lors de l'utilisation d'une stratégie de verrouillage PESSIMISTIC ou OPTIMISTIC, un gestionnaire de verrous est créé pour la mappe de sauvegarde. Il utilise une mappe de hachage pour rechercher les entrées verrouillées par une ou plusieurs transactions. S'il existe plusieurs entrées de mappe dans la mappe de hachage, plus nombreux sont les compartiments de verrouillage, meilleures sont les performances. Le risque de conflit de synchronisation Java™ diminue lorsque le nombre de compartiments augmente. Plus les compartiments de verrouillage sont nombreux, plus les accès simultanés le sont également. Les exemples précédents montrent comment une application peut définir le nombre de compartiments de verrouillage à utiliser pour une instance BackingMap donnée.

Java .NET

Verrouillage pessimiste

La stratégie de verrouillage PESSIMISTIC obtient des verrous pour les entrées de cache et doit être utilisée lorsque les données sont modifiées fréquemment. A chaque lecture d'une entrée de cache, un verrou est obtenu et maintenu de façon conditionnelle jusqu'à la fin de la transaction. La durée de certains verrous peut être paramétrée à l'aide des niveaux d'isolement de transaction pour la session.

La stratégie de verrouillage pessimiste est à utiliser pour les opérations de mappe en lecture et en écriture lorsqu'aucune autre stratégie de verrouillage n'est possible. Lorsqu'une mappe ObjectGrid est configurée avec la stratégie de verrouillage pessimiste, un verrou de transaction pessimiste est obtenu pour une entrée de mappe par la première transaction qui obtient cette entrée à partir de la mappe de sauvegarde. Le verrou pessimiste est maintenu jusqu'à la fin de la transaction. La stratégie de verrouillage pessimiste est généralement utilisée dans les cas suivants :

- Lorsque la mappe de sauvegarde est configurée et que les informations de gestion des versions ne sont pas disponibles.

  Restriction : Les mappes de sauvegarde configurées avec un plug-in de chargeur peuvent lire dans la mappe lors d'une transaction multipartition, mais pas y écrire.

- Lorsque la mappe de sauvegarde est utilisée directement par une application qui nécessite l'assistance de WebSphere eXtreme Scale Client pour le contrôle des accès simultanés.
- Lorsque les informations de gestion des versions sont disponibles mais que les transactions de mise à jour entrent régulièrement en conflit avec les entrées de sauvegarde, ce qui entraîne des échecs de mise à jour optimiste.

La stratégie de verrouillage pessimiste a une incidence considérable sur les performances et l'évolutivité. Par conséquent, réservez son utilisation aux opérations de lecture et d'écriture lorsqu'aucune autre stratégie de verrouillage n'est viable. Par exemple, en cas d'échecs réguliers des mises à jour optimistes ou de reprise après échec optimiste difficile à gérer pour une application.

Lorsque vous utilisez le verrouillage pessimiste, vous pouvez utiliser les méthodes de verrouillage pour

verrouiller des données ou des clés sans renvoyer de valeurs de données. Pour connaître la liste de ces méthodes et les types de verrou qu'elles obtiennent, voir Types de verrou.

Java

Verrouillage optimiste

La stratégie de verrouillage par défaut est OPTIMISTIC. Utilisez le verrouillage optimiste lorsque les données sont rarement modifiées. Les verrous ne sont maintenus que pour un laps de temps limité, c'est-à-dire pendant que les données sont lues à partir du cache et copiées dans la transaction. Lorsque le cache de transaction est synchronisé avec le cache principal, tous les objets mis en cache qui ont été mis à jour sont comparés à la version d'origine. Si la comparaison échoue, la transaction est annulée et une exception OptimisticCollisionException est provoquée.

La stratégie de verrouillage optimiste présuppose qu'il n'est pas possible que deux transactions tentent d'actualiser la même entrée de mappe au même moment. Le verrou n'est pas maintenu pendant toute la durée de la transaction car il est peu probable qu'une autre transaction puisse mettre à jour simultanément la même entrée de mappe. La stratégie de verrouillage optimiste est généralement utilisée dans les situations suivantes :

- Lorsqu'une mappe de sauvegarde est configurée et que les informations sur les versions sont disponibles.

  Restriction : Les mappes de sauvegarde configurées avec un plug-in de chargeur peuvent lire dans la mappe lors d'une transaction multipartition, mais pas y écrire.

- Lorsqu'une mappe de sauvegarde contient essentiellement des transactions qui exécutent des opérations de lecture. Les opérations insert, update ou remove sur les entrées de mappe ne sont pas fréquentes pour la mappe de sauvegarde.
- Lorsqu'une mappe de sauvegarde est insérée, mise à jour ou supprimée plus fréquemment qu'elle n'est lue mais que les transactions entrent rarement en conflit pour la même entrée de mappe.

A l'instar de la stratégie de verrouillage pessimiste, les méthodes de l'interface ObjectMap déterminent comment WebSphere eXtreme Scale Client tente automatiquement d'obtenir un mode de verrouillage pour l'entrée de mappe qui fait l'objet d'un accès. Toutefois, les stratégies pessimiste et optimiste diffèrent sur les points suivants :

- Comme dans la stratégie de verrouillage pessimiste, un mode de verrouillage S est obtenu par les méthodes get et getAll lorsque la méthode est appelée. En revanche, avec le verrouillage optimiste, le mode de verrouillage S n'est pas maintenu jusqu'à la fin de la transaction. Il est libéré avant que la méthode ne rende le contrôle à l'application. L'objectif de l'obtention d'un mode de verrouillage est que WebSphere eXtreme Scale Client vérifie que seules les données validées provenant d'autres transactions soient visibles pour la transaction en cours. Lorsque WebSphere eXtreme Scale Client a vérifié que les données ont été validées, le mode de verrouillage S est libéré. Au moment de la validation, une vérification optimiste des versions est effectuée pour s'assurer qu'aucune autre transaction n'a modifié l'entrée de mappe après la libération du mode de verrouillage S par la transaction en cours. Si une entrée n'est pas extraite de la mappe avant sa mise à jour, son invalidation ou sa suppression, le module d'exécution de WebSphere eXtreme Scale Client extrait implicitement l'entrée de la mappe. Cette opération get implicite est effectuée pour obtenir la valeur en cours au moment de la demande de modification de l'entrée.
- Contrairement à la stratégie de verrouillage pessimiste, les méthodes getForUpdate et getAllForUpdate sont traitées exactement comme les méthodes get et getAll de la stratégie de verrouillage optimiste. Un mode de verrouillage S est obtenu au début de la méthode et est libéré avant le renvoi du contrôle à l'application.

Toutes les autres méthodes ObjectMap sont traitées exactement comme dans le cas de la stratégie de verrouillage pessimiste. Lorsque la méthode commit est appelée, un mode de verrouillage X est obtenu pour toutes les entrées de mappe insérées, mises à jour, supprimées, corrigées ou invalidées, et le mode de verrouillage X est maintenu jusqu'à ce que la transaction termine le traitement de la validation.

La stratégie de verrouillage optimiste considère qu'aucune transaction simultanée ne tente de mettre à jour la même entrée de mappe. En partant de ce principe, il n'est pas nécessaire de maintenir le mode de verrouillage pendant toute la durée de la transaction, car il est peu probable que plusieurs transactions puissent mettre à jour simultanément la même entrée de mappe. Toutefois, étant donné qu'aucun mode de verrouillage n'est maintenu, une autre transaction simultanée peut potentiellement mettre à jour l'entrée de mappe après la libération du mode de verrou S par la transaction en cours.

Pour parer à cette éventualité, WebSphere eXtreme Scale Client obtient un verrou X lors de la validation et effectue une vérification optimiste des versions pour s'assurer qu'aucune autre transaction n'a modifié l'entrée de mappe après que la transaction en cours l'a lue dans la mappe de sauvegarde. Si une autre transaction modifie l'entrée de mappe, la vérification des versions échoue et une exception OptimisticCollisionException se produit. Cette exception force l'annulation de la transaction en cours et

l'application doit retenter la transaction tout entière. La stratégie de verrouillage optimiste s'avère utile lors qu'une mappe est principalement lue et que des mises à jour de la même entrée de mappe sont peu probables.

Java

Aucun verrouillage

Si le verrouillage n'est pas obligatoire car les données ne sont jamais mises à jour ou le sont au cours de période calmes, vous pouvez le désactiver en utilisant la stratégie de verrouillage NONE. Cette stratégie est très rapide car aucun gestionnaire de verrou n'est requis. Elle est idéale pour les tables de recherche et les mappes en lecture seule.

Lorsqu'une mappe de sauvegarde est configurée pour n'utiliser aucune stratégie de verrouillage, aucun verrou de transaction n'est obtenu pour une entrée de mappe.

Restriction : Les mappes de sauvegarde configurées pour utiliser une stratégie de non-verrouillage ne peuvent pas participer à une transaction multipartition.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Types de verrou

Lorsque vous utilisez les verrouillages pessimiste et optimiste, des verrous partagés (S), pouvant être mis à niveau (U) et exclusifs (X) sont utilisés pour maintenir la cohérence. Les verrous optimistes n'étant pas maintenus, c'est lorsque vous configurez le verrouillage pessimiste que les effets du verrouillage sont le plus apparents. Les verrous ont des cycles de vie. Les différents types de verrou ne sont pas tous compatibles entre eux de la même manière. Les verrous doivent être traités dans l'ordre approprié pour éviter les situations d'interblocage.

Verrous partagés, pouvant être mis à niveau et exclusifs

Lorsqu'une application appelle une méthode de l'interface de programmation de mappe, WebSphere eXtreme Scale Client tente automatiquement d'obtenir un verrou pour l'entrée de mappe qui fait l'objet d'un accès.

Java Dans le cas des applications Java, les verrous sont également obtenus lorsqu'une application utilise une méthode de recherche sur un index ou lance une requête.

Lorsque vous utilisez le verrouillage pessimiste, vous pouvez utiliser les méthodes de verrouillage pour verrouiller des données ou des clés sans renvoyer de valeurs de données. Ces méthodes vous permettent de verrouiller la clé dans la grille, ou de verrouiller la clé et de déterminer si la valeur existe dans la grille.

LockMode est une énumération qui vous permet d'indiquer les clés que vous souhaitez verrouiller à l'aide des valeurs possibles suivantes :

- Java Partagé (SHARED), pouvant être mis à niveau (UPGRADEABLE) et exclusif (EXCLUSIVE)
- .NET Partagé (Shared), pouvant être mis à niveau (Upgradeable) et exclusif (Exclusive)

WebSphere eXtreme Scale Client utilise les modes de verrouillage suivants en fonction de la méthode que l'application appelle dans l'interface de programmation de mappe :

Verrouillage S

Mode de verrouillage partagé pour la clé d'une entrée de mappe. La durée pendant laquelle le verrou S est maintenu dépend du niveau d'isolement de transaction utilisé. Avec le mode de verrouillage S, les transactions qui tentent d'obtenir un verrou S ou un verrou U (pouvant être mis à niveau) sur une même clé bénéficient d'un accès simultané, tandis que les autres transactions qui tentent d'obtenir un verrou X (exclusif) sur cette même clé sont bloquées.

Verrouillage U

Mode de verrouillage pouvant être mis à niveau pour la clé d'une entrée de mappe. Le verrou U est maintenu jusqu'à la fin de la transaction. Avec le mode de verrouillage U, les transactions qui obtiennent un verrou S sur une même clé bénéficient d'un accès simultané, tandis que les autres transactions qui tentent d'obtenir un verrou U ou un verrou X sur cette même clé sont bloquées.

Verrouillage X

Mode de verrouillage exclusif pour la clé d'une entrée de mappe. Le verrou X est maintenu jusqu'à la fin de la transaction. Avec le mode de verrouillage X, une seule transaction peut insérer, mettre à jour ou supprimer une entrée de mappe d'une valeur de clé donnée. Un verrou X bloque toutes les autres transactions qui tentent d'obtenir un verrou S, U ou X sur cette même clé.

Le mode de verrouillage S est plus faible que le mode de verrouillage U car il permet à davantage de transactions d'accéder simultanément à la même entrée de mappe. Le mode de verrouillage U est légèrement plus fort que le mode de verrouillage S car il bloque les autres transactions qui demandent un mode de verrouillage U ou X. Le mode de verrouillage S bloque uniquement les autres transactions qui demandent un mode de verrouillage X. Cette petite différence est pourtant importante lorsqu'il s'agit d'empêcher l'occurrence de certains interblocages. Le mode de verrouillage X est le plus fort car il bloque toutes les autres transactions qui tentent d'obtenir un verrou S, U ou X sur l'entrée de mappe concernée. Il garantit qu'une seule transaction peut insérer, mettre à jour ou supprimer une entrée de mappe, et empêche la perte des mise à jour lorsque plusieurs transactions tentent de mettre à jour la même entrée.

Consultez le tableau ci-dessous pour comprendre la relation entre ces modes de verrouillage et le comportement des méthodes existantes :

Tableau 1. Modes de verrouillage et méthodes équivalentes

Modes de verrouillage (Java / .NET)	Méthode Java équivalente	Méthode .NET équivalente
SHARED / Shared (partagé)	Méthodes get et getAll de l'interface ObjectMap, méthodes d'index et requêtes.	Méthodes Get(), GetAndLock(Key,LockMode. Shared), GetAndLockAll(KeyList, LockMode.Shared), GetAll.

UPGRADABLE / Upgradable (pouvant être mis à niveau)	getForUpdate(), getAllForUpdate()	GetAndLock(Key, LockMode.Updgradable), GetAndLockAll(KeyList, LockMode.Upgradable)
EXCLUSIVE / Exclusive (exclusif)	Méthodes getNextKey(), commit(), put, putAll, remove, removeAll, insert, update et touch ; méthodes invalidate et invalidateAll globales. (Aucun verrou n'est obtenu pour les méthodes invalidate et invalidateAll locales, car aucune des entrées de la mappe de sauvegarde (BackingMap) n'est invalidée par les appels à la méthode d'invalidation locale.) Java **Remarque :** Les méthodes upsert et upsertAll remplacent les méthodes put et putAll d'ObjectMap. Utilisez la méthode upsert pour indiquer à la mappe de sauvegarde et au chargeur qu'une entrée dans la grille de données doit placer la clé et la valeur dans la grille. La mappe de sauvegarde et le chargeur exécutent une insertion ou une mise à jour pour placer la valeur dans la grille et le chargeur. Si vous exécutez l'API upsert dans vos applications, le chargeur reçoit un type LogElement UPSERT, ce qui permet aux chargeurs d'exécuter des appels de fusion (merge) de base de données ou upsert au lieu d'insert ou update.	Commit(), Add, AddAll, Put, PutAll, Remove, RemoveAll, Replace, ReplaceAll, Touch, TouchAll, Invalidate, InvalidateAll **Remarque :** Les méthodes Put et PutAll sont équivalentes aux méthodes Java upsert et upsertAll.

Le tableau ci-dessous est une matrice de compatibilité des modes de verrouillage, qui résume les modes de verrouillage décrits et permet de déterminer la compatibilité entre les différents modes. Comment lire cette matrice : la première colonne indique le mode de verrouillage déjà octroyé. Pour chaque colonne suivante, l'en-tête indique le mode de verrouillage demandé par une autre transaction. Si une case contient `Oui`, le mode de verrouillage demandé par l'autre transaction est octroyé car il est compatible avec le mode de verrouillage déjà octroyé. Si une case contient `Non`, le mode de verrouillage demandé n'est pas octroyé, car il n'est pas compatible. Dans ce cas, l'autre transaction doit attendre que la première transaction libère le verrou qu'elle détient.

Tableau 2. Matrice de compatibilité des modes verrouillage

Verrou	**Verrou S (partagé)**	**Verrou U (pouvant être mis à niveau)**	**Verrou X (exclusif)**	**Force**
S (partagé)	Oui	Oui	Non	le plus faible
U (pouvant être mis à niveau)	Oui	Non	Non	normal
X (exclusif)	Non	Non	Non	le plus fort

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Interblocages

Un interblocage peut se produire lorsque deux transactions tentent de mettre à jour la même entrée de cache.

Exemple classique d'interblocage

Examinez la séquence de demandes de mode de verrouillage suivante :

1. Verrou X octroyé à la transaction 1 pour clé1.
2. Verrou X octroyé à la transaction 2 pour clé2.
3. Verrou X demandé par la transaction 1 pour clé2. (La transaction 1 est bloquée dans l'attente du verrou détenu par la transaction 2.)
4. Verrou X demandé par la transaction 2 pour clé1. (La transaction 2 est bloquée dans l'attente du verrou détenu par la transaction 1.)

La séquence précédente est l'exemple type de l'interblocage de deux transactions qui tentent d'acquérir plusieurs verrous dans un ordre différent. Pour éviter cet interblocage, chaque transaction doit obtenir ces verrous dans le même ordre.

Java

Prévention des interblocages grâce au verrouillage optimiste

Si la stratégie de verrouillage OPTIMISTE est utilisée et que la méthode flush sur l'interface ObjectMap n'est jamais utilisée par l'application, les modes de verrouillage ne sont demandés par la transaction que lors du cycle de validation. Pendant le cycle de validation, WebSphere eXtreme Scale Client détermine les clés pour les entrées de mappe qui doivent être verrouillées et demande les modes de verrouillage dans l'ordre des clés (comportement déterministe). Avec cette méthode, WebSphere eXtreme Scale Client évite la plupart des interblocages classiques.

Toutefois, eXtreme Scale n'empêche pas et ne peut pas empêcher tous les interblocages. Quelques scénarios doivent être gérés par l'application. Voici les cas dont l'application doit tenir compte et dans lesquels elle doit prendre des mesures préventives :

Il existe un cas dans lequel WebSphere eXtreme Scale Client est capable de détecter un interblocage sans être obligé d'attendre l'expiration du délai d'attente du verrou. Si ce cas se produit, une exception com.ibm.websphere.objectgrid.LockDeadlockExceptionexception est émise. Examinez cet exemple de code :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
sess.begin();
Person p = (IPerson)person.get("Lynn");
// Lynn had a birthday; so make her 1 year older.
p.setAge( p.getAge() + 1 );
person.put( "Lynn", p );
sess.commit();
```

Dans le même scénario, vous pouvez utiliser la méthode upsert dans l'exemple de code suivant :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
sess.begin();
Person p = (IPerson)person.get("Lynn");
// Lynn had a birthday; so make her 1 year older.
p.setAge( p.getAge() + 1 );
person.upsert( "Lynn", p );
sess.commit();
```

Dans cette situation, les deux transactions tentent de mettre à jour l'âge de l'objet personne Lynn. Ces deux transactions possèdent un verrou S sur l'entrée Lynn de la mappe PERSON suite à l'appel de la méthode person.get("Lynn"). Suite à l'appel de la méthode person.put ("Lynn", p), ces deux transactions tentent de transformer le verrou S en verrou X. Elles sont donc toutes les deux bloquées en attendant que l'autre transaction libère le verrou S qu'elle détient. Par conséquent, un interblocage se produit car un état d'attente circulaire existe entre les deux transactions. Un état d'attente circulaire existe lorsque plusieurs transactions tentent de transformer un verrou faible en verrou fort pour la même entrée de mappe. Dans ce scénario, une exception LockDeadlockException est émise au lieu d'une exception LockTimeoutException.

Dans les applications Java, l'application peut empêcher l'émission de l'exception LockDeadlockException dans l'exemple précédent en utilisant la stratégie de verrouillage optimiste au lieu de la stratégie de verrouillage pessimiste. La stratégie de verrouillage optimiste constitue la solution privilégiée lorsque la

mappe est essentiellement lue et que les mises à jour ne sont pas fréquentes.

Java .NET

Prévention des interblocages grâce au verrouillage pessimiste

.NET **Avertissement :** Les applications .NET ne prennent en charge que le verrouillage pessimiste. La suite de cette rubrique fait référence aux noms de méthode Java. Toutefois, les explications sont également valables pour les noms de méthode .NET. Ces méthodes sont les suivantes : Get, GetAndLock, GetAndLockAll, Put, Add, Replace et Remove.

Pour éviter les interblocages lorsque vous utilisez la stratégie de verrouillage pessimiste, procédez comme suit :

- Utilisez le niveau d'isolement de transaction `READ_COMMITTED` (lecture validée). Ce niveau empêche le verrou S acquis par la méthode get d'être maintenu jusqu'à la fin de la transaction. Si la clé n'est jamais invalidée dans le cache transactionnel, les lectures reproductibles sont toujours garanties.
- Utilisez d'autres méthodes d'obtention que la méthode get.
  - Java Utilisez la méthode getForUpdate.
  - .NET Utilisez la méthode GetAndLock ou GetAndLockAll.

  La première transaction qui appelle la méthode getForUpdate obtient un verrou U et non un verrou S. Avec ce mode de verrouillage, la deuxième transaction est bloquée lorsqu'elle appelle la méthode getForUpdate. En effet, le verrou U n'est octroyé qu'à une seule transaction. La deuxième transaction étant bloquée, elle ne possède aucun verrou sur l'entré de mappe. La première transaction n'est pas bloquée lorsqu'elle tente de mettre à niveau le mode de verrouillage U vers le mode de verrouillage X suite à l'appel de la méthode put. Voilà pourquoi on dit du mode de verrouillage U qu'il peut être mis à niveau. Lorsque la première transaction est terminée, la deuxième est débloquée et le verrou U lui est octroyé. Pour empêcher ce scénario d'interblocage de la mise à niveau du mode de verrouillage, une application peut employer la méthode getForUpdate au lieu de la méthode get en cas d'utilisation de la stratégie de verrouillage pessimiste.

  Important : Cette solution n'empêche pas les transactions en lecture seule de lire une entrée de mappe. Ces transactions appellent la méthode get, mais jamais les méthodes put, insert, update et remove. Les accès simultanés sont aussi nombreux que lors de l'utilisation de la méthode get classique. Ils ne sont réduits que lorsque la méthode getForUpdate est appelée par plusieurs transactions pour la même entrée de mappe.

  Lorsque des transactions appellent la méthode getForUpdate pour plusieurs entrées de mappe, vous devez le savoir pour être sûr que les verrous U sont acquis dans le même ordre par chaque transaction. Par exemple, supposons que la première transaction appelle la méthode pour la clé 1 et la clé 2, et qu'une autre transaction simultanée appelle la méthode pour les mêmes clés mais dans l'ordre inverse. Cette séquence entraîne l'interblocage classique car plusieurs verrous sont obtenus dans un ordre différent par des transactions différentes. Afin d'éviter tout interblocage, l'application doit toujours s'assurer que chaque transaction accède à plusieurs entrées de mappe dans l'ordre des clés. Etant donné que le verrou U est obtenu lors de l'appel de la méthode getForUpdate et non au moment de la validation, WebSphere eXtreme Scale Client ne peut pas ordonner les demandes de verrouillage de la même manière que lors du cycle de validation. Dans ce cas, c'est l'application qui doit contrôler l'ordre de verrouillage.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Accès aux données et transactions

WebSphere eXtreme Scale Client utilise des transactions. Dès lors qu'une application dispose d'une connexion à une grille de données, vous pouvez accéder aux données et interagir avec celles-ci dans la grille.

Java

Transactions dans les applications Java

Avec les applications Java, vous pouvez établir une connexion client avec une instance répartie.

Lorsqu'une application interagit avec une Session, elle doit se trouver dans le contexte d'une transaction. Une transaction est initiée, puis validée ou annulée, en appelant les méthodes Session.begin, Session.commit et Session.rollback sur l'objet Session. Les applications peuvent également utiliser le mode de validation automatique, dans lequel la Session initie et valide automatiquement une transaction lorsque l'application interagit avec des mappes. Le mode de validation automatique est toutefois plus lent.

Dans une application Java, chaque unité d'exécution doit avoir sa propre Session. Lorsque vous souhaitez que l'application utilise ObjectGrid sur une unité d'exécution, appelez l'une des méthodes getSession pour obtenir une session. Lorsque l'application n'utilise plus la session, utilisez la méthode Session.close(). Cette méthode ferme la session, la renvoie au pool et libère ses ressources. La fermeture d'une session est facultative, mais elle améliore les performances des appels suivants vers la méthode getSession(). Si l'application utilise une infrastructure d'injection de dépendance telle que Spring, vous pouvez injecter une Session dans le bean d'une application lorsque cela est nécessaire.

Lorsque vous avez obtenu une Session, l'application peut accéder aux données stockées dans des mappes de la grille d'objets. L'API fondée sur les mappes s'obtient à l'aide de la méthode Session.getMap.

.NET

Transactions dans les applications .NET

Dans les applications .NET, chaque unité d'exécution doit posséder son propre objet IGridMapPessimisticTx ou IGridMapPessimisticAutoTx. Avec l'objet IGridMapPessimisticTx, vous utilisez la propriété Transaction pour commencer, valider ou annuler explicitement la transaction. Avec l'objet IGridMapPessimisticAutoTx, ces trois opérations ont lieu automatiquement. Dès que vous avez obtenu l'un de ces objets, l'application peut accéder aux données stockées dans la grille de données.

Logique d'utilisation des transactions

Les transactions peuvent paraître lentes, mais vous devez les utiliser pour les raisons suivantes :

1. Pour pouvoir annuler des modifications en cas d'exception ou si la logique métier requiert l'annulation d'une modification d'état.
2. Pour placer des verrous sur les données et les libérer tout au long de la durée de vie d'une transaction, afin que les modifications soient apportées de manière atomique (toutes les modifications sont appliquées, ou aucune).
3. Pour générer une unité atomique de réplication.
4. Java Pour mettre à jour plusieurs partitions.

Vous pouvez personnaliser le niveau de prise en charge des transactions nécessaire. Votre application peut désactiver la prise en charge de l'annulation et le verrouillage, mais cela sera à ses dépens. En effet, elle devra alors gérer elle-même l'absence de ces fonctions. Voici des exemples de la manière dont une application peut gérer la prise en charge des transactions :

- Java Une application peut désactiver le verrouillage en configurant la stratégie de verrouillage NONE pour la mappe dynamique. Pour plus d'informations, voir Création de mappes dynamiques avec les API Java. Cette stratégie permet de gagner en rapidité, mais plusieurs transactions simultanées peuvent désormais modifier les mêmes données sans protection les unes des autres. L'application est responsable du verrouillage et de la cohérence des données lorsque NONE est utilisé. Cette option n'est pas valide pour les applications WebSphere eXtreme Scale Client for .NET, qui ne prennent en charge que la stratégie de verrouillage PESSIMISTIC.

Java Si l'application modifie un objet récupéré à l'aide de la valeur CopyMode NONE, elle modifie directement la copie validée de cet objet. Dans ce mode, l'annulation de la transaction n'a aucune signification. Vous modifiez la seule copie dans l'ObjectGrid. Bien que l'utilisation de CopyMode NONE offre une grande rapidité, vous devez en mesurer les conséquences. Une application utilisant ce mode ne doit jamais annuler la transaction. Si elle le fait, les modifications ne sont pas reportées dans l'index *et* ne sont pas répliquées si la réplication est activée.

Les valeurs par défaut sont plus simples à utiliser et risquent moins d'entraîner des d'erreurs. Si vous favorisez les performances au détriment de la fiabilité des données, l'application doit savoir ce qu'elle fait afin d'éviter tout problème.

Transactions et partitions

Les transactions des applications Java peuvent mettre à jour une ou plusieurs partitions, mais par défaut une seule partition est mise à jour. Les applications .NET ne peuvent mettre à jour qu'une seule partition.

Utilisez l'API de session TxCommitProtocol pour activer le support de transaction multipartition pour WebSphere eXtreme Scale Client. Vous pouvez utiliser les deux options suivantes :

- TxCommitProtocol.ONEPHASE (par défaut) : les transactions d'un client peuvent lire plusieurs partitions mais ne peuvent en mettre à jour qu'une seule. Les tentatives de mise à jour de plusieurs partitions échouent.
- TxCommitProtocol.TWOPHASE : les transactions d'un client peuvent lire et mettre à jour plusieurs partitions. Elles utilisent le protocole de validation en deux phases pour que les données écrites dans les partitions soient automatiquement validées ou annulées. Si une transaction écrit dans une seule partition, le protocole de validation en une phase est utilisé.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:

Java Développement d'applications pour écrire dans des transactions multipartitions pour WebSphere eXtreme Scale dans un environnement autonome

Java Interaction avec les données dans une transaction pour les applications Java

.NET Interaction avec les données dans une transaction pour les applications .NET

Isolement des transactions

Vous pouvez utiliser l'un des trois niveaux d'isolement de transaction suivants afin d'optimiser la sémantique de verrouillage qui maintient la cohérence dans chaque mappe de cache : lecture reproductible, lecture validée et lecture non validée.

Présentation de l'isolement de transaction

L'isolement de transaction définit comment les modifications apportées par une opération deviennent visibles aux autres opérations simultanées.

Vous pouvez définir l'un des trois niveaux d'isolement de transaction suivants afin d'optimiser la sémantique de verrouillage que WebSphere eXtreme Scale Client utilise pour maintenir la cohérence dans chaque mappe de cache : lecture reproductible, lecture validée et lecture non validée.

Vous pouvez définir le niveau d'isolement de transaction de l'une des manières suivantes :

- Java Dans l'interface Session avec la méthode setTransactionIsolation. L'isolement de transaction peut être modifié à tout moment pendant la durée de vie de la session si une transaction n'est pas en cours d'exécution.
- .NET Dans l'interface IGridTransaction avec la propriété TransactionIsolationLevel.

Le produit f=garantit le respect des divers aspects de la sémantique d'isolement de transaction en paramétrant la demande et le maintien des verrous (S) partagés. L'isolement de transaction n'a aucune incidence sur les mappes configurées pour utiliser la stratégie de verrouillage optimiste ou aucune stratégie de verrouillage, ou lorsque des verrous pouvant être mis à jour (U) sont obtenus.

Lecture reproductible avec verrouillage pessimiste

Le niveau d'isolement de transaction Lecture reproductible est le niveau par défaut. Ce niveau d'isolement empêche les lectures de pages modifiées et les lectures non reproductibles mais pas les lectures fantômes. Une lecture de pages modifiées est une opération de lecture de données qui ont été modifiées par une transaction mais qui n'ont pas été validées. Une lecture non reproductible peut survenir lorsqu'aucun verrou en lecture n'a été obtenu lors de l'opération de lecture. Une lecture fantôme a lieu lorsque deux opérations de lecture identiques ont été effectuées mais que deux jeux de résultats ont été renvoyés en raison d'une mise à jour intervenue entre les opérations de lecture. Dans les applications Java, les lectures fantômes sont possibles lorsque vous utilisez des requêtes ou des index, car les verrous ne sont pas obtenus pour des plages de données mais uniquement pour les entrées de cache qui correspondent à l'index ou aux critères de requête. Le produit obtient une lecture reproductible en maintenant les verrous S jusqu'à la fin de la transaction qui détient le verrou concerné. Etant donné qu'un verrou X n'est octroyé que lorsque tous les verrous S sont libérés, toutes les transactions maintenant un verrou S obtiendront obligatoirement la même valeur lors de la relecture.

Lecture validée avec verrouillage pessimiste

Le niveau d'isolement de transaction de lecture validée peut être utilisé avec WebSphere eXtreme Scale Client, ce qui empêche les lectures de pages modifiées mais pas les lectures non reproductibles ni les lectures fantômes ; WebSphere eXtreme Scale Client continue ainsi à utiliser des verrous S pour lire les données de la mappe de cache mais libère immédiatement ces verrous.

Lecture non validée avec verrouillage pessimiste

Le niveau d'isolement de transaction lecture non validée peut être utilisé avec WebSphere eXtreme Scale Client, ce qui autorise les lectures de pages modifiées, les lectures non reproductibles et les lectures fantômes.

Rubrique parent : Présentation du traitement des transactions

Référence associée:
Java Exemples Java pour l'isolement de transaction

Java

Validation en deux phases et reprise sur incident

2.5+ Le protocole de validation en deux phases coordonne toutes les partitions qui participent à une transaction répartie, en déterminant si la transaction doit être validée ou annulée.

Dans une grille de données répartie, les partitions sont réparties sur plusieurs machines virtuelles Java™. Ces machines JVM peuvent se trouver sur plusieurs systèmes. Une transaction qui écrit dans plusieurs partitions peut impliquer des décisions transactionnelles qui ont une incidence sur plusieurs systèmes. Lorsque la transaction est validée à l'aide d'un protocole de validation en deux phases, ce processus de validation conserve l'ensemble de la transaction ou ne la conserve pas du tout. Le processus de validation en deux phases garantit ce résultat même en présence d'erreurs liées au système, aux partitions ou à la communication. Si un incident survient au cours de la deuxième phase, le client WebSphere eXtreme Scale tente de résoudre le problème automatiquement, sauf si l'erreur répond à certains critères pour lesquels vous pouvez intervenir manuellement.

Une transaction prévue pour écrire dans plusieurs partitions utilise le protocole de validation en deux phases. Ce protocole garantit que le processus de validation est cohérent dans tous les partitions et systèmes. WebSphere eXtreme Scale fait office de coordinateur qui contrôle le processus de validation en deux phases. Les partitions impliquées dans la transaction sont appelées participants ou gestionnaires de ressources (RM). Au cours de la deuxième phase du protocole de validation, le coordinateur délègue l'une des partitions pour faire office de gestionnaire de transactions (TM). Le gestionnaire de transactions assure le suivi de la décision concernant chaque transaction et la reprise de la transaction en cas de problème.

Première phase :

Lorsqu'une application valide une transaction, le client WebSphere eXtreme Scale démarre la première phase en envoyant une demande de préparation de validation à chaque partition identifiée comme gestionnaire de ressources. Chaque partition applique les modifications de transaction aux mappes de sauvegarde et conserve tous les verrous pour protéger l'intégrité des données. Le gestionnaire de ressources notifie le client WebSphere eXtreme Scale. Une fois que toutes les partitions identifiées comme gestionnaire de ressources ont répondu avec succès, le client WebSphere eXtreme Scale lance la seconde phase du protocole de validation.

Seconde phase :

Si une ou plusieurs partitions échouent au cours de la première étape, le coordinateur annule (invalide) toutes les partitions au cours de la deuxième phase. En revanche, si toutes les partitions RM répondent avec succès, le client WebSphere eXtreme Scale délègue l'une des partitions pour faire office de partition TM. En tant que coordinateur, WebSphere eXtreme Scale lance la seconde phase du protocole de validation en envoyant une demande de validation ou d'annulation à toutes les partitions impliquées dans la transaction. Chaque partition identifiée comme RM applique les modifications à la mappe de sauvegarde ou les annule, puis lève tous les verrous. Le gestionnaire de ressources RM notifie ensuite le client WebSphere eXtreme Scale. Si une ou plusieurs partitions ont échoué au cours de la seconde phase, la partition TM déléguée reprend automatiquement la transaction. Cette reprise automatique garantit que toutes les partitions impliquées dans la transaction sont cohérentes.

Phase d'attente de validation :

La phase d'attente de validation est la période entre le traitement réussi de la première phase par la partition RM et le début de la seconde phase. Pendant cette période, la partition RM ne sait pas si elle doit valider ou annuler la transaction. La partition RM maintient les verrous. Dans ce cas, les autres transactions peuvent être affectées par une augmentation des conflits de verrouillage.

Reprise après incident au cours d'une validation en deux phases

En cas d'erreur au cours de la première phase, le client WebSphere eXtreme Scale annule la transaction. Si l'une des partitions ne parvient pas à valider la transaction, le gestionnaire de transactions retente régulièrement de valider la transaction. Voici un exemple des messages qui sont consignés dans le journal dans ce scénario :

```
00000099 TransactionLog I CWOBJ8705I: La résolution automatique de la transaction WXS-
40000139-DF01-216D-E002-1CB456931719 sur RM:TestGrid:TestSet2:20 attend toujours une
décision. Une nouvelle tentative de résolution de la transaction sera exécutée dans 30
secondes.
```

Laissez le client WebSphere eXtreme Scale résoudre la transaction. N'intervenez manuellement que si la transaction n'est pas récupérée après une minute ou que l'application est affectée par un grand nombre de conflits de verrous parce qu'il s'agit d'une transaction en attente. Pour plus d'informations sur la reprise manuelle d'une transaction, voir Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition.

Rubrique parent : Présentation du traitement des transactions

Rubrique parent : 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Concepts associés:
Java Stratégies de verrouillage

Tâches associées:
Java Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition
Java Résolution des exceptions de délai d'attente de verrouillage

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7 Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE "EN L'ETAT" SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM® Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des

logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances, ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Tous les tarifs indiqués sont les prix de vente actuels suggérés par IBM et sont susceptibles d'être modifiés sans préavis. Les tarifs appliqués peuvent varier selon les revendeurs.

Ces informations sont fournies uniquement à titre de planification. Elles sont susceptibles d'être modifiées avant la mise à disposition des produits décrits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les exemples de programmes sont fournis "en l'état", sans garantie d'aucune sorte. IBM ne sera en aucun cas responsable de tout dommage résultant de l'utilisation des exemples de programmes.

Toute copie totale ou partielle de ces exemples de programmes et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit :

© nom de votre société) (année). Des segments de code sont dérivés des exemples de programmes d'IBM Corp.

© Copyright IBM Corp. _entrez l'année ou les années_. All rights reserved.

Documentation sur l'interface de programmation

Cette publication contient principalement des informations qui ne sont destinées à être utilisées comme interfaces de programmation de WebSphere eXtreme Scale. Il décrit également des interfaces de programmation qui permettent au Client d'écrire des programmes pouvant utiliser les services de WebSphere eXtreme Scale. Ces informations sont identifiées par un texte d'introduction dans un chapitre ou une section ou par le repère suivant : Informations relatives aux interfaces de programmation.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à l'adresse www.ibm.com/legal/copytrade.shtml.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Remarques sur les règles de confidentialité

Les produits IBM® Software ("Offres logicielles"), dont les solutions de type SaaS (logiciel sous forme de services), peuvent utiliser des cookies ou d'autres technologies pour collecter des informations sur l'utilisation des produits, afin notamment d'améliorer l'expérience des utilisateurs finaux ou encore de personnaliser les interactions avec ces derniers. Dans de nombreux cas, ces Offres logicielles ne collectent pas d'informations permettant d'identifier l'utilisateur. Certaines d'entre elles peuvent toutefois vous aider à collecter ce type d'informations. Si la présente Offre logicielle utilise des cookies pour collecter des informations identifiant la personne, des informations spécifiques sur l'utilisation des cookies par cette offre figurent ci-après.

Cette Offre logicielle n'utilise pas de cookies ni d'autre technologie pour collecter des informations permettant d'identifier l'utilisateur.

Si les configurations déployées pour cette Offre logicielle vous permettent en tant que client de collecter des informations identifiant la personne auprès des utilisateurs finaux, via des cookies et d'autres technologies, il vous est conseillé de vous renseigner auprès de votre conseil juridique sur les lois applicables à cette collecte de données et notamment aux exigences en matière de notification et d'accord.

Pour plus d'informations concernant l'utilisation de différentes technologies, et notamment des cookies, à ces fins, consultez la politique IBM de protection des renseignements personnels (http://www.ibm.com/privacy) et la déclaration Online Privacy Statement d'IBM (http://www.ibm.com/privacy/details/us/en), et particulièrement les sections intitulées "Cookies, Web Beacons and Other Technologies" et "Software Products and Software-as-a Service".

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Scénarios

Le scénario utilise des informations du monde réel pour construire une image complète. Exécutez un scénario pour comprendre nouveaux les concepts ou pour accomplir des tâches communes.

Scénario : Configuration d'une grille de données d'entreprise
Configurez une grille de données d'entreprise lorsque vous voulez connecter des applicationsJava™ et .NET à la même grille de données.

2.5+ **Sécurisation de WebSphere DataPower XC10 Appliance**
Les grilles de données du dispositif stockent des informations qui sont confidentielles et doivent être protégées.

Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Vous pouvez migrer une session de réplication de mémoire à mémoire ou une session de base de données pour utiliser la gestion de session WebSphere eXtreme Scale.

Scénario : Configuration d'une grille de données d'entreprise

Configurez une grille de données d'entreprise lorsque vous voulez connecter des applicationsJava™ et .NET à la même grille de données.

Avant de commencer

- Installez le produit. Pour les clients, vous pouvez utiliser des clients Java et .NET. Pour plus d'informations, voir Installation de WebSphere DataPower XC10 Appliance.
- Si vous effectuez une mise à niveau à partir d'une version précédente, tous les serveurs de conteneur et de catalogue doivent être au même niveau d'édition. Pour plus d'informations, voir Mise à jour de WebSphere DataPower XC10 Appliance.

1. Présentation de la grille de données d'entreprise
   Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Avec le nouveau transport et le nouveau format de sérialisation, vous pouvez connecter des clients Java et .NET à une même grille de données.
2. Configuration d'IBM eXtremeIO (XIO)
   IBM® eXtremeIO (XIO) est un mécanisme de transport qui remplace ORB (Object Request Broker).
3. Développement d'applications de grille de données d'entreprise
   Après avoir configuré IBM eXtremeIO, vous pouvez écrire des applications qui accèdent à la grille de données d'entreprise.

Rubrique parent : 2.5+ Scénarios

Configuration d'IBM eXtremeIO (XIO)

2.5+ IBM® eXtremeIO (XIO) est un mécanisme de transport qui remplace ORB (Object Request Broker).

Avant de commencer

- 2.5 Pour configurer XIO, vous devez posséder WebSphere eXtreme Scale Client version 8.6.0.2 ou ultérieur.

Vous pouvez configurer XIO pour tous les serveurs de conteneur du domaine de service de catalogue en activant XIO dans les serveurs de catalogue. Les serveurs de conteneur reconnaissent le type de transport du serveur de catalogue et utilisent ce type de transport.

2.5

Pourquoi et quand exécuter cette tâche

XIO est le mécanisme de transport par défaut si vous créez une nouvelle configuration avec la version 2.5 ou suivante. Si vous effectuez une mise à niveau depuis une version précédente du microprogramme, votre paramètre de transport est défini pour utiliser ORB.

Procédure

Activez le mécanisme de transport XIO dans la collectivité. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Paramètres** > **Services de communication**. Sélectionnez le mécanisme de transport XIO. Le paramètre de transport est un paramètre de niveau collectivité. Par conséquent, si vous mettez à jour le paramètre de transport, tous les dispositifs de la collectivité doivent être redémarrés.

Résultats

Les serveurs que vous avez configurés utilisent le transport XIO. Pour vérifier que la configuration est correcte, voir Affichage du type de transport du domaine de service de catalogue.

La collectivité utilise le mécanisme de transport XIO. Lorsque vous activez XIO, vous activez également :

eXtreme Data Format (XDF)

XDF sérialise et stocke les clés et les valeurs dans la grille de données dans un format indépendant du langage. Si vous utilisez XDF, les applications Java et .NET peuvent accéder aux mêmes objets de grille de données.

IBM eXtremeMemory

eXtremeMemory vous aide à éviter les pauses de récupération d'espace, ce qui permet stabiliser les performances et de bénéficier de temps de réponse plus prévisible.

2.5+ **Affichage du type de transport du domaine de service de catalogue**
Vous pouvez afficher le type de transport qui est actuellement utilisé pour le domaine de service de catalogue.

Rubrique parent : Scénario : Configuration d'une grille de données d'entreprise
Rubrique précédente : Présentation de la grille de données d'entreprise
Rubrique suivante : Développement d'applications de grille de données d'entreprise
Rubrique parent : Configuration des collectivités et des zones
Rubrique précédente : Configuration d'une réplication multimaître entre les collectivités

Développement d'applications de grille de données d'entreprise

Après avoir configuré IBM eXtremeIO, vous pouvez écrire des applications qui accèdent à la grille de données d'entreprise.

Avant de commencer

- Vous devez déjà disposer d'applications Java ou .NET qui accèdent à la grille de données. Pour plus d'informations sur l'écriture d'applications, voir Module 2 du guide d'initiation : Création d'une application client.

Extension de classe
Le format de données XDF (eXtreme data format) permet d'étendre les classes. Avec l'extension de classe, vous pouvez faire évoluer les définitions de classe utilisées dans la grille de données sans affecter les anciennes applications qui utilisent les version précédentes d'une classe. Ces anciennes classes sont des données d'accès qui se trouvent dans la même mappe que les nouvelles applications.

Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET
Utilisez des annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes Java et .NET.

Mappage de clés avec des partitions avec des annotations PartitionKey
Un alias PartitionKey est utilisé pour identifier les zones ou les attributs sur lesquels un calcul de code de hachage est exécuté pour déterminer la partition dans laquelle les données sont sauvegardées. L'annotation PartitionKey n'est valide que sur les attributs de clé.

Types de données équivalents entre Java et C#
Lorsque vous développez des applications de grille de données d'entreprise, les types de données entre les applications Java et C# doivent être compatibles.

Rubrique parent : Scénario : Configuration d'une grille de données d'entreprise
Rubrique précédente : Configuration d'IBM eXtremeIO (XIO)

Extension de classe

Le format de données XDF (eXtreme data format) permet d'étendre les classes. Avec l'extension de classe, vous pouvez faire évoluer les définitions de classe utilisées dans la grille de données sans affecter les anciennes applications qui utilisent les version précédentes d'une classe. Ces anciennes classes sont des données d'accès qui se trouvent dans la même mappe que les nouvelles applications.

Présentation

L'extension de classe est une autre extension de l'identification des classes et des zones qui détermine si deux types sont suffisamment compatibles pour fonctionner conjointement. Les classes peuvent fonctionner ensemble lorsqu'une des classes dispose d'un nombre de zones inférieur à l'autre classe. Les scénarios utilisateur suivants sont intégrés dans l'implémentation :

Plusieurs versions d'une même classe d'objets

Dans ce scénario, vous disposez d'une mappe dans une application de vente qui permet de suivre les clients. Cette mappe dispose de deux interfaces. La première est dédiée aux achats en ligne et la seconde, aux achats effectués par téléphone. Dans la version 2 de cette application de vente, vous décidez d'accorder une réduction sur les achats en ligne en fonction des habitudes d'achat des clients. Cette réduction est stockée avec l'objet Client. Les employés de la vente par téléphone utilisent toujours la version 1 de l'application qui ne sait pas qu'il existe une nouvelle zone de réduction dans la version en ligne. Vous voulez que les objets Client de la version 2 de l'application fonctionnent avec les objets Client créés avec la version 1 de l'application et vice versa.

Plusieurs versions d'une classe d'objets différente

Dans ce scénario, vous disposez d'une application de vente écrite en Java™ qui conserve une mappe des objets Client. Vous disposez également d'une autre application écrite en C# qui permet de gérer l'inventaire de l'entrepôt et qui expédie les commandes des clients. Ces classes sont actuellement compatibles en fonction des noms des classes, des zones et des types. Dans l'application de vente Java, vous voulez ajouter une option à l'enregistrement Client pour associer le vendeur à un compte de client. Cependant, vous ne voulez pas mettre à jour l'application d'entrepôt pour stocker cette zone, car elle est inutile dans l'entrepôt.

Plusieurs versions incompatibles d'une même classe

Dans ce scénario, les applications de vente et d'inventaire contiennent toutes les deux un objet Client. L'application d'inventaire utilise une zone d'ID qui correspond à une chaîne et l'application de vente, une zone d'ID qui est un entier. Ces types ne sont pas compatibles. Par conséquent, les objets ne sont probablement pas stockés dans la même mappe. Les objets doivent être gérés par la sérialisation XDF et traités comme deux types distincts. Bien que ce scénario n'entre pas réellement dans le cadre de l'extension de classe, il doit être pris en compte dans la conception générale de l'application.

Détermination pour l'extension

XDF tente d'étendre une classe lorsque les noms de classe correspondent et que les noms des zones ne génèrent pas de conflits de zones. Les annotations ClassAlias et FieldAlias sont utiles lorsque vous voulez faire correspondre des classes entre des applications C# et Java dans lesquelles les noms des classes ou les zones diffèrent légèrement. Vous pouvez placer ces annotations dans l'application Java ou C# ou les deux applications. Cependant, la recherche de la classe dans l'application Java peut s'avérer moins efficace que de définir ClassAlias dans l'application C#. Pour plus d'informations sur les annotations ClassAlias et FieldAlias, voir Annotations ClassAlias et FieldAlias

Impact des zones manquantes dans les données sérialisées

Le constructeur de la classe n'est pas appelé au cours de la sérialisation. Par conséquent, les zones manquantes ont une valeur par défaut qui lui est affectée en fonction du langage. L'application qui ajoute de nouvelles zones doit pouvoir détecter les zones manquantes et réagir lorsqu'une ancienne version de la classe est extraite.

Pour que les anciennes applications conservent les nouvelles zones, la seule solution consiste à mettre à jour les données.

Une application peut exécuter une extraction et mettre à jour la mappe avec une ancienne version de la classe qui ne contient pas certaines zones dans la valeur sérialisée depuis le client. Le serveur fusionne les valeurs sur le serveur et détermine si des zones de la version d'origine sont fusionnées dans le nouvel enregistrement. Si une application exécute une extraction, puis supprime et insère une entrée, les zones de la valeur d'origine sont perdues.

Fusion des fonctions

Les options dans une matrice ou une collecte ne sont pas fusionnées par XDF. Il n'est pas toujours aisé de déterminer si une mise à jour dans une matrice ou une collecte doit changer les éléments de lamatrice ou du type. Si une fusion se produit en fonction du positionnement, lorsqu'une entrée dans la matrice est déplacée, XDF peut fusionner les zones qui doivent être associées. Par conséquent, XDF ne tente pas de fusionner le

contenu des matrices ou des collectes. Cependant, si vous ajoutez une matrice dans une nouvelle version d'une définition de classe, la matrice est de nouveau fusionnée dans la version précédente de la classe.

Rubrique parent : Développement d'applications de grille de données d'entreprise

Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Utilisez des annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes Java™ et .NET.

Avant de commencer

- Vous devez avoir configuré IBM® eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).

Pourquoi et quand exécuter cette tâche

Vous pouvez envisager d'utiliser des annotations ClassAlias et FieldAlias si vous disposez d'une classe Java et voulez créer une classe C# correspondante. Dans ce scénario, vous pouvez ajouter des annotations à la classe C#, qui contiennent le nom de classe Java. Pour plus d'informations sur les annotations ClassAlias et FieldAlias, voir Annotations ClassAlias et FieldAlias.

Procédure

Utilisez des annotations ClassAlias et FieldAlias pour corréler des objets entre une classe Java et une classe C#. Java

Figure 1. Exemple Java avec des annotations ClassAlias et FieldAlias

```
@ClassAlias("Employee")
class com.company.department.Employee {

     @FieldAlias("id")
     int myId;

      String name;
}
```

.NET

Figure 2. Exemple .NET avec des attributs ClassAlias et FieldAlias

```
[ ClassAlias( "Employee") ]
class Com.MyCompany.Employee {

     [ FieldAlias("id") ]
      int identifier;

      string name;
}
```

Annotations ClassAlias et FieldAlias
Utilisez les annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes. Vous pouvez partager des données entre deux classes Java, ou entre une classe Java et une classe .NET.

.NET **Annotations ClassAlias et FieldAlias**
Utilisez les annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes. Vous pouvez partager des données entre deux classes Java, ou entre une classe Java et une classe .NET.

Rubrique parent : Développement d'applications de grille de données d'entreprise
Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Concepts associés:
Annotations ClassAlias et FieldAlias

Référence associée:
Fichier de propriétés du client

Information associée:
Leçon 2.3 : Création d'une application de grille de données

Annotations ClassAlias et FieldAlias

Utilisez les annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes. Vous pouvez partager des données entre deux classes Java™, ou entre une classe Java et une classe .NET.

Si vous définissez deux classes avec le même nom et zones, les données de la grille de données sont automatiquement partagées entre les classes. Par exemple, si vous disposez d'une classe Client1 dans votre application Java et d'une classe Client1 dans votre application .NET contenant les mêmes zones, les données sont partagées entre les classes. Cet exemple suppose que le nom de classe contient également le qualificateur de classe, qui est également le nom de package dans Java, et l'espace-noms en C#. Le nom du package et de l'espace-noms sont automatiquement partagés car ces noms correspondent. Voir l'exemple suivant, dans lequel les deux noms sont insensibles à la casse :

```
Java:
package com.mycompany.app
public class SampleClass {
int field1;
String field2;
}

C#
namespace Com.MyCompany.App
public class SampleClass {
int field1;
string field2;
}
```

Cependant, vous pouvez également corréler des données entre des classes ayant des noms différents. Pour corréler les données à stocker dans la grille de données entre des classes portant des noms différents, utilisez des annotations ClassAlias ou FieldAlias.

Entre deux applications Java : vous pouvez définir deux classes avec des noms différents dans des environnements d'application Java distincts. En marquant les classes avec la même annotation ClassAlias, toutes les zones et tous les types de zone sont en correspondance entre ces deux classes. Les classes sont corrélées avec le même ID de type de classe même si elles portent des noms de classe différents. Le même ID de type de classe et les métadonnées peuvent être réutilisés ensuite dans des environnements d'exécution d'application Java différents.

Entre une application Java et une application .NET : vous pouvez utiliser des annotations similaires dans une application C# pour corréler la classe C# avec une classe Java. Les attributs ClassAlias définis pour la classe C# et les zones sont mis en correspondance avec une classe Java ayant la même annotation ClassAlias.

Rubrique parent : Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET
Rubrique parent : .NET Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Tâches associées:
Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Référence associée:
Fichier de propriétés du client

Information associée:
Leçon 2.3 : Création d'une application de grille de données

Mappage de clés avec des partitions avec des annotations PartitionKey

Un alias PartitionKey est utilisé pour identifier les zones ou les attributs sur lesquels un calcul de code de hachage est exécuté pour déterminer la partition dans laquelle les données sont sauvegardées. L'annotation PartitionKey n'est valide que sur les attributs de clé.

Avant de commencer

Vous devez utiliser eXtreme Data Format (XDF). XDF est activé lorsque vous utilisez IBM eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).

Pourquoi et quand exécuter cette tâche

Vous définissez un alias PartitionKey pour que plusieurs classes enregistrent les données dans la même partition. Par exemple, si vous définissez la valeur PartitionKey comme clé departmentID, les enregistrements d'employé sont placés dans la même partition.

L'interface PartionableKey est l'interface Java existante et elle est prioritaire sur l'annotation PartitionableKey dans C#.

Procédure

- Java Définissez des annotations PartitionKey dans une zone dans une application Java.

  ```
  class Employee {
      int empId;

      @PartitionKey(order = 0)
      int deptId;

         }
  ```

  Vous pouvez définir des annotations PartitionKey sur plusieurs clés ou l'alias PartitionKey dans une classe. Pour d'autres exemples sur la définition des annotations PartitionKey dans les applications Java, voir Documentation des API Java : type d'annotation PartitionKeys.

- .NET Définissez des attributs PartitionKey dans une zone dans une application .NET.

  ```
  class Employee {
      int empId;

      [PartitionKey]
      int deptId;
  }
  ```

  Vous pouvez également définir des attributs PartitionKey dans des classes .NET. Pour plus d'informations, voir Documentation des API .NET : classe PartitionKeyAttribute.

Rubrique parent : Développement d'applications de grille de données d'entreprise
Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Types de données équivalents entre Java et C#

Lorsque vous développez des applications de grille de données d'entreprise, les types de données entre les applications Java et C# doivent être compatibles.

Tableau 1. Types de données équivalents entre Java et C#

Type Java	Type C#
boolean	bool
java.lang.Boolean	bool?
byte	sbyte or byte
java.lang.Byte	sbyte?
short	short?, ushort
java.lang.Short	short?, ushort?
int	int, uint, ushort
java.lang.Integer	int?, uint?
long	long, ulong, uint
java.lang.Long	long?, ulong?, uint?
short ou int	ushort
java.lang.Short ou java.lang.Integer	ushort?
int ou long	uint
java.lang.Integer ou java.lang.Long	uint?
long ou BigInteger	ulong
java.lang.Long ou java.lang.BigInteger	ulong?
char, java.lang.Character	char
java.lang.Character	char?
float, java.lang.Float	float
java.lang.Foat	float?
double	double
java.lang.Double	double?
java.math.BigDecimal	decimal or decimal?
java.math.BigInteger	decimal, long or ulong?
java.lang.String	string
java.util.Date, java.util.Calendar	System.DateTime
java.util.Date(rounding), java.util.Calendar(rounding)	System.DateTime
java.util.ArrayList	System.Collections.ArrayList, System.Collections.Generic.List
java.util.HashMap	System.Collections.Generic.Dictionary, System.Collections.Hashtable
java.util.LinkedList	System.Collections.Generic.LinkedList
java.util.ArrayList, java.util.Vector	System.Collections.Generic.List
java.util.Stack	System.Collections.Generic.Stack
java.util.Vector	System.Collections.ArrayList, System.Collections.Generic.List

Rubrique parent : Développement d'applications de grille de données d'entreprise

Sécurisation de WebSphere DataPower XC10 Appliance

Les grilles de données du dispositif stockent des informations qui sont confidentielles et doivent être protégées.

Avant de commencer

Certains aspects de ce scénario, tels que l'activation de la norme FIPS (Federal Information Processing Standard) 140-2, nécessitent que tous les membres d'une collectivité soient au niveau le plus récent. Si le dispositif à sécuriser fait partie d'une collectivité, le microprogramme de tous les membres de cette collectivité doit être mis à niveau pour que vous puissiez effectuer les tâches de ce scénario.

Pourquoi et quand exécuter cette tâche

WebSphere DataPower XC10 Appliance comprend des contrôles de sécurité globaux. La configuration par défaut comporte des mots de passe par défaut, des clés SSL et des valeurs d'authentification confidentielles que vous devez modifier. Suivez ce scénario pour modifier la configuration, puis procédez au déploiement du dispositif sécurisé.

Pour un déploiement sécurisé, utilisez plusieurs couches de protection pour optimiser la sécurité. Le premier élément de protection consiste à utiliser des pare-feu pour segmenter le réseau. Le modèle standard à niveaux pour les applications Web est constitué des clients, d'un niveau présentation constitué de serveurs HTTP, d'un niveau application constitué des serveurs d'application, d'un niveau données et d'un niveau stockage.

Les dispositifs WebSphere DataPower XC10 Appliance sont déployés dans le niveau données. En règle générale, il convient de placer les serveurs de la couche présentation dans une zone démilitarisée (DMZ) protégée par un pare-feu, et de placer les niveaux application, données et stockage dans des segments du réseau protégés par d'autres pare-feu. Ne déployez pas un dispositif dans une zone démilitarisée. Vous devez protéger vos dispositifs comme tous les autres éléments du niveau données, conformément aux bonnes pratiques communément utilisées.

Cependant, pour optimiser la protection contre les menaces à la sécurité, utilisez un mécanisme de défense efficace, comprenant des mesures supplémentaires qui protègent le fonctionnement des dispositifs et les données stockées dans la grille de données. Ces mesures supplémentaires offrent une protection contre les attaques externes, mais interdisent également les accès non autorisés aux données par les employés et les sous-traitants qui pourraient avoir accès aux segments du réseau contenant les dispositifs.

Les différentes étapes de ce scénario sont effectuées dans la console Web de WebSphere DataPower XC10 Appliance. Elles peuvent également être automatisées en appelant l'interface de ligne de commande HTTP à partir d'un programme. Pour plus d'informations concernant l'interface de ligne de commande HTTP, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

1. 2.5+ Configuration d'un accès sécurisé à la grille de données
   Configurez un accès administrateur et spécifiez des paramètres pour vos collectivités de dispositifs afin de configurer une authentification et une autorisation sécurisées pour l'accès à la grille de données.

2. 2.5+ Activation de la sécurité pour les grilles de données
   Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

3. 2.5+ Configuration de la sécurité du client
   Une fois que vous avez sécurisé la grille de données sur le dispositif, vous devez configurer les clients afin qu'ils puissent se connecter à la grille de données sécurisée.

Rubrique parent : 2.5+ Scénarios

Configuration d'un accès sécurisé à la grille de données

Configurez un accès administrateur et spécifiez des paramètres pour vos collectivités de dispositifs afin de configurer une authentification et une autorisation sécurisées pour l'accès à la grille de données.

Pourquoi et quand exécuter cette tâche

Pour une protection optimale contre les menaces pour la sécurité, suivez la procédure ci-dessous afin de réduire les menaces externes et les accès non autorisés aux données par les employés et les sous-traitants qui pourraient avoir accès aux segments du réseau dans lesquels les dispositifs communiquent avec les clients et les serveurs.

Avertissement : Dans la console d'administration, lorsque vous spécifiez les paramètres de collectivité pour la communication de serveur à serveur et la protection des données sur le réseau, vous pouvez configurer les paramètres qui contrôlent la sécurité. Si vous modifiez ces paramètres, vous devez redémarrer l'ensemble de la collectivité. Pour pouvoir définir ces paramètres, cliquez sur **Collectivité** > **Paramètres**. Pour plus d'efficacité, effectuez toutes les modifications en une seule opération afin de ne devoir redémarrer la collectivité qu'une seule fois.

Procédure

1. Sécurisez l'accès administrateur

   Pour pouvoir configurer le dispositif, utilisez l'ID administrateur `xcadmin` et le mot de passe d'administration par défaut `xcadmin`. Cet ID utilisateur et ce mot de passe donnent un accès complet à l'ensemble des fonctions d'administration et des données du dispositif et de la collectivité. Il est donc très important de configurer un mot de passe d'administration racine qui soit difficile à deviner. Pour ce faire, dans la console Web, cliquez sur **Collectivité** > **Utilisateurs**. Sélectionnez l'utilisateur **Administrateur** et modifiez son mot de passe.

2. Configurez la sécurité du transport SSL

   Le dispositif est configuré avec un fichier de clés et un fichier de clés certifiées par défaut. Le fichier de clés certifiées par défaut inclut le certificat de signataire du fichier de clés par défaut. Etant donné que ce certificat est livré avec chaque dispositif, il doit être remplacé pour que la configuration SSL soit sécurisée. Le remplacement du certificat se déroule en plusieurs étapes.

   a. Créez un nouveau certificat et une clé privée Ce certificat peut être autosigné, mais il est préférable qu'il soit émis par une autorité de certification locale.
   b. Créez ou modifiez le fichier de clés certifiées Une méthode consiste à télécharger le fichier de clés certifiées par défaut à partir du dispositif afin de supprimer le certificat de signataire du fichier de clés certifiées libellé **IBM WebSphere DataPower XC10**, puis à ajouter un certificat afin d'établir une relation de confiance avec le nouveau fichier de clés. Si le nouveau fichier de clés comporte un certificat autosigné, ce certificat doit être importé dans le fichier de clés certifiées. Si le nouveau fichier de clés comporte un certificat émis par une autorité de certification locale, le certificat racine de cette autorité doit être ajouté au fichier de clés certifiées.
   c. Téléchargez les nouveaux fichiers de clés et de clés certifiées.
   d. Modifiez le paramètre d'alias de certificat afin qu'il corresponde au nom du certificat dans le fichier de clés. Les fichiers de clés certifiées de tous les clients de grille de données doivent également être mis à jour pour établir une relation de confiance avec le certificat du dispositif.

      Gérez le fichier de clés et le fichier de clés certifiées à l'aide de la commande keytool fournie avec l'environnement d'exécution Java (JRE, Java Runtime Environment). Vous pouvez aussi utiliser d'autres outils de gestion de clés. Reportez-vous aux tâches administratives ci-après, qui permettent de gérer les fichier de clés et les fichiers de clés certifiées.

      - Générez un certificat autosigné.

        ```
        keytool -genkey -alias ogsample -keystore key.jks -storetype JKS -keyalg
        rsa -dname "CN=ogsample, OU=OGSample, O=acme, L=Your City, S=Your State,
        C=Your Country" -storepass ogpass
        -keypass ogpass -validity 3650
        ```

      - Supprimez le certificat de signataire par défaut du dispositif du fichier de clés certifiées téléchargé à partir du dispositif.

        ```
        keytool -delete -alias “ibm websphere datapower xc10” -keystore trust.jks
        ```

```
-storetype JKS -keypass xc10pass
```

- Exportez le nouveau certificat du fichier de clés.

  ```
  keytool -export -alias ogsample -keystore key.jks -file temp.key -
  storepass ogpass
  ```

- Importez le certificat dans le fichier de clés certifiées.

  ```
  keytool -import -noprompt -alias ogsamplepublic -keystore trust.jks -file
  temp.key -storepass xc10pass
  ```

  Cette étape d'importation est nécessaire pour configurer le fichier de clés certifiées afin qu'il puisse établir une relation de confiance avec un certificat exporté à partir d'un fichier de clés. Cette étape est effectuée dans plusieurs contextes. Le fichier de clés certifiées du client doit être mis à jour pour qu'une relation de confiance puisse être établie avec le certificat du fichier de clés du dispositif, et, dans certaines configurations, le fichier de clés certifiées du dispositif doit être mis à jour pour qu'une relation de confiance puisse être établie avec le certificat du fichier de clés du client. L'idéal serait que les certificats du fichier de clés du client et du dispositif soient émis par une autorité de certification locale, auquel cas la confiance sera établie par l'importation du certificat de signataire racine de cette autorité de certification.
  - Modifiez le mot de passe dans le fichier de clés et le fichier de clés certifiées.

    ```
    keytool -storepasswd -new newpass -keystore trust.jks -storepass xc10pass
    ```

  - Téléchargez les nouveaux fichiers de clés et de clés certifiées vers la collectivité de dispositifs.

e. Spécifiez un mode SSL parmi les options suivantes : **TCP/IP**, où SSL n'est pas utilisé pour la grille de données de communication, **TLS pris en charge** et **TLS requis**.

   TLS requis est recommandé pour la sécurité. Si TLS n'est pas utilisé, les mots de passe et les données de grille sensibles sont transmises non chiffrées sur les liaisons réseau qui relient les clients de la grille et le dispositif.

   Remarque : Lorsque la communication TLS est obligatoire, le fichier de clés certifiées du dispositif doit être configuré pour l'établissement d'une relation de confiance avec le certificat contenu dans le fichier de clés du client, et le fichier de clés certifiées du client doit être configuré pour l'établissement d'une relation de confiance avec le certificat contenu dans le fichier de clés du serveur. Voir Configuration de la sécurité du client.

f. Activez la norme FIPS (Federal Information Processing Standard).

   Vous pouvez configurer la collectivité de dispositifs pour qu'elle utilise la norme FIPS 140-2 pour toutes les communications réseau chiffrées. Cette norme garantit une protection élevée des données transmises. Sélectionnez **l'option d'activation de la cryptographie FIPS 140-2** pour activer FIPS.

   Certaines versions de navigateur Web ne fonctionnent pas avec un serveur sur lequel FIPS est activée. Toutefois, les versions actuelles de la plupart des navigateurs (y compris Mozilla Firefox, Microsoft Internet Explorer et Google Chrome) prennent en charge la communication avec ce type de serveur. Vous devrez peut-être configurer votre navigateur de manière à activer TLS, car sslv3 n'est pas pris en charge en mode FIPS. Pour plus d'informations, voir la documentation de votre navigateur.

g. Activez l'authentification par certificat client.

   Lorsque cette option est activée, le fichier de clés de chaque client et navigateur qui communique avec le dispositif doit être configuré pour comporter un certificat accepté par le fichier de clés certifiées du dispositif. L'authentification par certificat client n'est pas utilisée pour le transport ORB. Elle n'est utilisée que pour le transport HTTPS et XIO. Il n'est pas nécessaire d'activer l'authentification par certificat client pour avoir une configuration sécurisée.

3. Configurez l'authentification de serveur à serveur.

   La communication de grille de données entre les dispositif d'une collectivité, et entre des domaines de dispositif liés, est authentifiée à l'aide d'une clé secrète partagée. Les dispositifs sont déjà configurés avec une clé secrète par défaut codée en dur. Vous devez modifier cette clé secrète afin de disposer d'une configuration sécurisée. Pour pouvoir spécifier une clé secrète unique, sélectionnez **Remplacer la clé secrète d'authentification définie par défaut en usine (recommandé)**.

Cette clé secrète doit être une phrase de passe longue et difficile à deviner. Notez-la et stockez-la dans un endroit sûr. Lorsque les collectivités sont jointes dans des domaines liés, chaque collectivité doit être configurée avec la même clé secrète.

4. Exigez l'authentification pour toutes les requêtes concernant la grille de données.

   Vous pouvez configurer l'authentification pour chaque requête client à la grille de données. Par défaut, cette authentification n'est pas obligatoire. Toutefois, vous pouvez protéger l'accès aux grilles de données en sécurisant chaque grille individuellement, et sécuriser la configuration en exigeant l'authentification pour toutes les requêtes aux grilles. Lorsque cette option est définie, chaque client doit être configuré avec un ID utilisateur et un mot de passe connus de la collectivité de dispositifs, ou, dans le cas de l'authentification LDAP, enregistrés dans LDAP. Seul l'administrateur racine (ID utilisateur xcadmin) peut se connecter sans passer par l'authentification LDAP.

   Une fois effectuées les étapes 2 à 4, soumettez les modifications pour redémarrer la collectivité. Les nouvelles valeurs sont automatiquement propagées aux dispositifs qui doivent être assimilés dans la collectivité. Si vous avez activé la norme FIPS, elle est activée sur chaque dispositif avant son assimilation dans la collectivité.

5. Désactivez l'accès Telnet.

   Le configuration par défaut du dispositif comprend un serveur Telnet actif. La communication Telnet du dispositif ne prend pas en charge SSL. Pour sécuriser la configuration, désactivez le serveur Telnet. Pour ce faire, établissez une session SSH avec le dispositif en utilisant l'ID utilisateur et le mot de passe de l'administrateur, puis lancez la commande **`platform service telnet disable`**. Cette commande n'agissant pas sur l'ensemble de la collectivité, vous devez l'exécuter sur chaque dispositif. La commande **`platform service telnet enable`** démarre Telnet s'il a été désactivé. Cette procédure est manuelle et ne peut être automatisée.

6. Configurez l'authentification LDAP.

   L'authentification pour le navigateur, pour REST et pour l'accès de la grille au dispositif peut s'effectuer de deux manières : vous pouvez enregistrer les identités authentifiées dans la collectivité de dispositifs ou dans LDAP. Vous pouvez utiliser l'une ou l'autre de ces deux méthodes dans une configuration sécurisée.

   Conseil : Notez que l'authentification de l'administrateur racine utilise toujours un mot de passe vérifié par la collectivité et non par LDAP.

   Lorsque l'authentification LDAP est utilisée, protégez la connexion LDAP par SSL afin que les mots de passe ne circulent pas en clair sur le réseau. Pour activer SSL pour une connexion LDAP, spécifiez une URL LDAPS, telle que `ldaps://ldapserver.company.com:636`. Maintenant, vous devez configurer le fichier de clés certifiées du dispositif avec un certificat afin d'établir une relation de confiance avec le serveur LDAP en vue de la communication SSL. Pour plus d'informations sur la configuration de l'authentification LDAP, voir Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP.

Rubrique parent : 2.5+ Sécurisation de WebSphere DataPower XC10 Appliance
Rubrique suivante : 2.5+ Activation de la sécurité pour les grilles de données

Information associée:
Configuration des fichiers JSSE conformes à la norme FIPS

Activation de la sécurité pour les grilles de données

Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

Pourquoi et quand exécuter cette tâche

Lorsqu'une grille de données appartient à une collectivité, vous pouvez configurer la sécurité et le contrôle d'accès pour cette grille en particulier. L'activation de la sécurité permet d'imposer aux clients qui tentent d'accéder à la grille de s'authentifier. Indépendamment de ce paramètre, si la collectivité est configurée pour exiger l'authentification pour tous les accès aux grilles de données, l'authentification est toujours exigée pour l'accès à cette grille spécifique. Mais exiger l'authentification peut ne pas restreindre suffisamment les accès aux grilles. Par exemple, dans le cas de l'authentification LDAP, n'importe quel utilisateur LDAP peut accéder à cette grille spécifique. Pour en restreindre l'accès, vous pouvez activer l'autorisation pour cette grille.

Important : Si vous modifiez les paramètres de sécurité d'une grille de données, cette dernière redémarre automatiquement. Lorsque la grille de données redémarre, les données qui s'y trouvent sont perdues. Par conséquent, configurez la sécurité de vos grilles de données avant de commencer à y sauvegarder des données.

La communication via la passerelle REST est toujours sécurisée, même si la sécurité n'est pas activée sur la grille de données. Pour plus d'informations, voir Passerelle REST : Configuration de la sécurité.

Procédure

1. Dans l'interface utilisateur, accédez aux paramètres de la grille de données. Cliquez sur **Grille de données** > ***type_grille_données***. Cliquez sur le nom de la ***nom_grille_données*** à modifier.
2. Activez la sécurité ou l'autorisation pour la grille de données. Cliquez sur **Activer la sécurité** pour permettre à tous les utilisateurs pouvant accéder à l'interface utilisateur d'accéder à la grille de données. Si vous souhaiter restreindre davantage l'accès, cliquez sur **Activer l'autorisation**.

   Remarque : Vous devez activer l'autorisation pour une grille de données de session.

   Une fois l'autorisation activée, vous pouvez spécifier une liste d'utilisateurs ou de groupe d'utilisateurs dans la liste**Accès accordé à**. Lorsque l'autorisation est activée, seuls les utilisateurs figurant dans cette liste sont autorisés à accéder aux données de la grille de données. Vous pouvez affecter les accès suivants aux utilisateurs ou aux groupes d'utilisateurs en cliquant sur le nom du type d'accès par défaut qui est affiché dans l'interface utilisateur :
   - **read** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire ou interroger des données de la grille de données.
   - **write** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire, interroger et écrire des données dans la grille de données.
   - **create** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire, interroger, écrire, insérer et créer des mappes dynamiques dans la grille de données.
   - **all** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire, interroger, écrire, insérer, créer des mappes dynamiques, supprimer et invalider des données de la grille de données. Les administrateurs de dispositif disposent du droit **all** par défaut.

   Lorsque vous changez les paramètres de sécurité et d'autorisation, le délai d'attente est de cinq minutes.
   - **Délai d'authentification** : si vous changez le mot de passe d'un utilisateur déjà authentifié dans la grille de données, les données d'identification d'origine sont toujours valides pendant 5 minutes.
   - **Délai d'autorisation** : si vous supprimez une autorisation pour un utilisateur, ce dernier reste autorisé pendant cinq minutes. Ce délai ne s'applique qu'aux autorisations supprimées. Si vous ajoutez une autorisation à un utilisateur, l'utilisateur est autorisé immédiatement.

Rubrique parent : 2.5+ Sécurisation de WebSphere DataPower XC10 Appliance
Rubrique précédente : 2.5+ Configuration d'un accès sécurisé à la grille de données
Rubrique suivante : 2.5+ Configuration de la sécurité du client
Rubrique parent : Configuration des grilles de données
Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Configuration de la sécurité du client

Une fois que vous avez sécurisé la grille de données sur le dispositif, vous devez configurer les clients afin qu'ils puissent se connecter à la grille de données sécurisée.

Pourquoi et quand exécuter cette tâche

Si le dispositif est configuré pour exiger TLS, le client doit être configuré pour le transport SSL et il doit avoir un fichier de clés et un fichier de clés certifiées appropriés. Si une authentification est requise, le client doit également être configuré avec un ID utilisateur et un mot de passe. La procédure à suivre varie selon que l'installation du client s'exécute de façon autonome ou dans un processus WebSphere Application Server.

Dans le cas d'un environnement autonome, vous devez configurer un fichier `client.properties` contenant les paramètres à transmettre à l'application de grille de données. Si votre environnement inclut WebSphere Application Server, utilisez les outils de WebSphere Application Server pour configurer la sécurité du client.

Procédure

1. Si le client est une installation autonome, utilisez un fichier de propriétés du client pour configurer la sécurité du client et la communication entre le client et le serveur. Pour plus de détails concernant l'emplacement et le format de ce fichier, voir IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité.

   Pour connaître les propriétés de sécurité que vous pouvez configurer pour les clients Java™ ou .NET, voir Fichier de propriétés du client.

2. Si le client s'exécute dans un processus WebSphere Application Server, vous devez configurer les valeurs de configuration SSL (y compris le fichier de clés et le fichier de clés certifiées) à l'aide des outils de WebSphere Application Server.

   Un utilitaire fourni permet d'importer le certificat à partir du fichier de clés du dispositif dans le fichier de clés certifiées de WebSphere Application Server. Pour plus de détails, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

   Vous devez configurer ce fichier de clés certifiées de WebSphere Application Server pour qu'il accepte le certificat du fichier de clés du dispositif. Si le dispositif est configuré pour l'authentification par certificat client, ou si la communication via ORB (Object Request Broker) est utilisée, le fichier de clés certifiées du dispositif doit aussi être configuré pour accepter le certificat du fichier de clés de WebSphere Application Server.

   Si le dispositif est configuré pour exiger l'authentification, le client du dispositif qui s'exécute sous WebSphere Application Server doit aussi être configuré pour nécessiter une authentification à l'aide de données d'identification. Vous pouvez configurer le client avec un fichier de propriétés, comme pour les clients autonomes Java. Toutefois, lorsque le client du dispositif est installé sur WebSphere Application Server, la console d'administration de WebSphere Application Server est modifiée pour vous permettre d'entrer les données d'identification. Vous pouvez ainsi configurer les domaines de service de catalogue et spécifier les propriétés d'authentification du client.

3. Si le client s'exécute dans un environnement dans lequel la norme FIPS (Federal Information Processing Standard) est activée sur le serveur, il doit utiliser le protocole d'établissement de liaison TLS pour communiquer avec le dispositif. Il n'est pas possible dans ce cas d'utiliser le protocole d'établissement de liaison SSL. Le protocole TLS se configure en indiquant `protocol=TLS` dans le fichier de propriétés du client.

   - Lorsque XIO est utilisé et que le fichier de propriétés du client ne contient pas de paramètre de protocole, la valeur par défaut est SSL et TLS. Ce paramétrage fonctionne donc lorsque la norme FIPS est activée sur le dispositif.
   - Lorsque la communication de type ORB est utilisée sur un client autonome, le protocole doit être défini sur TLS dans le fichier de propriétés du client pour permettre la communication avec un dispositif en mode FIPS.
   - Lorsque la communication de type ORB est utilisée et que le client s'exécute avec WebSphere Application Server, ce sont les paramètres de sécurité de WebSphere qui sont utilisés pour définir le protocole d'établissement de liaison. Par défaut, il prend la valeur SSL_TLS, qui est appropriée lorsque la norme FIPS est activée sur le dispositif.

     Ce paramètre peut être configuré sur TLS ou SSL_TLS à l'aide de la console d'administration de WebSphere Application Server. Cliquez sur **Sécurité** > **Certificat SSL et gestion des clés** > **Gérer les configurations de sécurité des noeuds finals**. Sélectionnez un nom de cellule et cliquez sur **Configurations SSL** > **CellDefaultSSLSettings** > **Paramètres de la qualité de protection (QoP)**. Assurez-vous que la zone de protocole contient bien TLS ou SSL_TLS.

Rubrique parent : 2.5+ Sécurisation de WebSphere DataPower XC10 Appliance
Rubrique précédente : 2.5+ Activation de la sécurité pour les grilles de données

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Référence associée:
Fichier de propriétés du client

Java

Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Vous pouvez migrer une session de réplication de mémoire à mémoire ou une session de base de données pour utiliser la gestion de session WebSphere eXtreme Scale.

Avant de commencer

- Pour le support de session des applications client exécutées sur WebSphere Application Server dans le cluster, WebSphere eXtreme Scale doit être installé sur les déploiements de noeud WebSphere Application Server, y compris le noeud de gestionnaire de déploiement. Voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Les étapes dans ce scénario s'appliquent à la version 8.5 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction la version WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Dans le cadre de la migration vers une session WebSphere DataPower XC10 Appliance, vous devez noter les paramètres de configuration précédents dans la console d'administration WebSphere Application Server. Lors de la migration vers une session WebSphere DataPower XC10 Appliance, les paramètres de configuration doivent refléter ce que vous avez déjà configuré pour la base de données ou la session mémoire à mémoire.

Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Dans le cadre d'une migration vers une session HTTP dans la grille de données, vous devez créer un domaine de service de catalogue à l'aide de la console d'administration de WebSphere Application Server.

Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Vous devez utiliser les paramètres de configuration précédents que vous avez notés dans la console d'administration WebSphere Application Server pour associer une application ou un serveur d'applications à la gestion de session WebSphere eXtreme Scale.

Rubrique parent : 2.5+ Scénarios
Rubrique parent : Création de grilles de données de session

Tâches associées:
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

Java

Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server

Dans le cadre de la migration vers une session WebSphere DataPower XC10 Appliance, vous devez noter les paramètres de configuration précédents dans la console d'administration WebSphere Application Server. Lors de la migration vers une session WebSphere DataPower XC10 Appliance, les paramètres de configuration doivent refléter ce que vous avez déjà configuré pour la base de données ou la session mémoire à mémoire.

Pourquoi et quand exécuter cette tâche

La console d'administration WebSphere Application Server contient des paramètres spécifiques que vous devez noter. Vous en aurez besoin lors de la mise à jour du fichier `splicer.properties`. Les étapes de cette procédure s'appliquent à la version 8.5 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction de la version WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Client version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Procédure

1. Démarrez la console d'administration WebSphere Application Server.
    - Si vous avez déjà défini des paramètres au niveau du serveur, accédez à :
        a. **Serveurs > Types de serveur > Serveurs d'applications WebSphere**
        b. Dans la zone **Serveurs d'applications**, sélectionnez **le nom du serveur**
        c. Dans la zone **Paramètres de conteneur**, cliquez sur **Gestion de session**/
    - Si vous avez déjà défini des paramètres au niveau de l'application, accédez à :
        a. **Applications > Toutes les applications**.
        b. Dans la zone **Serveurs d'applications**, sélectionnez le **nom de l'application**.
        c. Dans la zone **Propriétés du module Web**, cliquez sur **Gestion des sessions**.
2. Dans les **propriétés générales**, cochez la case **d'autorisation de dépassement**.
3. Dans la zone des **propriétés générales**, notez les paramètres WebSphere Application Server. Vous en aurez besoin lors de la mise à jour les propriétés dans le fichier `splicer.properties`.

    Tableau 1. Paramètres de configuration pour mettre à jour le fichier splicer.properties

Paramètres dans la console d'administration WebSphere Application Server	Propriétés à mettre à jour dans le fichier splicer.properties
Activer les cookies	useCookies
Activer la réécriture des URL	useURLEncoding
Nombre maximal de sessions en mémoire	sessionTableSize

4. Dans la zone des **propriétés générales**, si la case **Activer les cookies** est sélectionnée, cliquez dessus et notez les paramètres WebSphere Application Server. Vous en aurez besoin pour mettre à jour les propriétés dans le fichier `splicer.properties`.

    Tableau 2. Paramètres de configuration dans le fichier splicer.properties

Paramètres dans la console d'administration WebSphere Application Server	Propriétés à mettre à jour dans le fichier splicer.properties
Domaine du cookie	cookieDomain
Chemin du cookie	cookiePath

5. Cliquez sur **Gestion des sessions**, puis dans la zone des **propriétés supplémentaires**, cliquez sur **Distributed environment settings**.
6. Dans la zone **Distributed Sessions** remplacez la base de données ou la configuration de réplication de mémoire à mémoire par **None**.
7. Cliquez sur **Custom Tuning Properties** et notez les paramètres WebSphere Application Server. Vous en aurez besoin pour mettre à jour les propriétés dans le fichier `splicer.properties`.

    Tableau 3. Paramètres de configuration des propriétés dans le fichier splicer.properties

Paramètres dans la console d'administration WebSphere Application Server	Propriétés à mettre à jour dans le fichier splicer.properties
Fréquence d'écriture	replicationInterval

Fréquence d'écriture	replicationInterval
Write contents	fragmentedSession

Que faire ensuite

Ensuite, créez le domaine de service de catalogue pour une session WebSphere DataPower XC10 Appliance.

Rubrique parent : Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

Java

Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données

Dans le cadre d'une migration vers une session HTTP dans la grille de données, vous devez créer un domaine de service de catalogue à l'aide de la console d'administration de WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Cette procédure s'applique à la version 8.5 de la console d'administration de WebSphere Application Server. Ces informations peuvent varier en fonction de la version de WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Client version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Créez le domaine de service de catalogue pour la grille de données à l'aide de la console d'administration de WebSphere Application Server. Pour plus d'informations, voir Création de domaines de service de catalogue dans WebSphere Application Server.

Procédure

1. Démarrez la console d'administration WebSphere Application Server.
2. Dans le menu supérieur, cliquez sur **Administration de système > WebSphere eXtreme Scale > Domaines de service de catalogue**

   Remarque : Si WebSphere eXtreme Scale n'apparaît pas, cela signifie que votre profil WebSphere Application Server n'a pas été étendu pour la grille de données. .
3. Cliquez sur **Nouveau**.
4. Définissez le nom du service de catalogue dans la zone **Nom**.
5. Dans la zone **Serveurs de catalogue**, choisissez **Serveur distant** et définissez l'emplacement ou le nom du serveur distant dans la zone.
6. Définissez le numéro de port dans la zone **Port d'écoute**.
7. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.

Que faire ensuite

Ensuite, utilisez les paramètres de configuration précédents que vous avez notés dans la console d'administration WebSphere Application Server pour associer une application ou un serveur d'applications à la gestion de session WebSphere eXtreme Scale.

Rubrique parent : Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

Java

Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents

Vous devez utiliser les paramètres de configuration précédents que vous avez notés dans la console d'administration WebSphere Application Server pour associer une application ou un serveur d'applications à la gestion de session WebSphere eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Cette procédure s'applique à la version 8.5 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction la version WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Client Version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Procédure

- Si vous voulez configurer une application pour l'associer à la gestion de session WebSphere eXtreme Scale, procédez comme suit :
    1. Démarrez la console d'administration WebSphere Application Server.
    2. Dans le menu supérieur, cliquez sur **Applications > Toutes les applications**.
    3. Dans la zone **Applications d'entreprise WebSphere**, sélectionnez le **nom d'application**.
    4. Dans la zone des propriétés **Module Web**, cliquez sur **Gestion de session**.
    5. Cliquez sur **Paramètres de gestion de session eXtreme Scale**.
    6. Si WebSphere eXtreme Scale n'apparaît pas, cela implique que votre profil WebSphere Application Server n'a pas été étendu pour WebSphere eXtreme Scale. Pour plus d'informations, voir .
    7. Pour configurer une application pour WebSphere eXtreme Scale Client, procédez comme suit :
        a. Dans la liste **>Gérer la persistance des sessions par**, sélectionnez **IBM WebSphere DataPower XC10 Appliance**
        b. Définissez l'adresse IP ou le nom d'hôte du domaine du service de catalogue que vous avez créé depuis la liste.
        c. Définissez un nom d'utilisateur et un mot de passe et testez la connexion.
        d. Définissez une nouvelle grille de données ou sélectionnez une grille de données existante.
    8. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
    9. Un fichier splicer.properties est créé pour l'application. L'emplacement du fichier `splicer.properties` est la valeur d'une nouvelle propriété `{application name},com.ibm.websphere.xs.sessionFilterProps`. Pour rechercher la propriété personnalisée, accédez à **Administration de système> Cellule** et cliquez sur **Propriétés personnalisées**.
    10. Mettez à jour le fichier `splicer.properties` avec les valeurs que vous avez obtenues dans Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server.
    11. Redémarrez les processus serveur d'applications.

    Remarque : Changez `splicer.properties` au niveau du gestionnaire de déploiement pour que les propriétés soient synchronisées sur l'agent de noeud. Si vous mettez à jour `splicer.properties` au niveau du noeud, le gestionnaire de déploiement remplace le fichier `splicer.properties` lors de la synchronisation suivante.

    Remarque : Si vous revenez dans la gestion de session de base de données et la gestion de session WebSphere eXtreme Scale, le fichier `splicer.properties` est recréé et les modifications que vous avez apportées sont remplacées. Pour plus d'informations sur le processus de synchronisation de fichier depuis le gestionnaire de déploiement vers les notes et connaître les éléments modifiés, voir Synchronisation des fichiers de gestion de système.

- Si vous voulez configurer un serveur d'applications pour l'associer à la gestion de session WebSphere eXtreme Scale, procédez comme suit :
    1. Démarrez la console d'administration WebSphere Application Server.
    2. Dans le menu supérieur, cliquez sur **Serveurs > Types de serveur > Serveurs d'applications WebSphere**.
    3. Dans la zone **Serveurs d'applications**, sélectionnez le **nom du serveur**.
    4. Dans la zone **Paramètres de conteneur**, cliquez sur **Gestion de session**/
    5. Cliquez sur **Paramètres de gestion des sessions eXtreme Scale**.

        Remarque : Si WebSphere eXtreme Scale ne s'affiche pas, cela implique que votre profil WebSphere Application Server n'a pas été étendu pour WebSphere eXtreme Scale. Pour plus d'informations, voir .

6. Pour configurer une application pour WebSphere eXtreme Scale Client, procédez comme suit :
    a. Dans la liste **>Gérer la persistance des sessions par**, sélectionnez **IBM WebSphere DataPower XC10 Appliance**
    b. Définissez l'adresse IP ou le nom d'hôte du domaine du service de catalogue que vous avez créé depuis la liste.
    c. Définissez un nom d'utilisateur et un mot de passe et testez la connexion.
    d. Définissez une nouvelle grille de données ou sélectionnez une grille de données existante.
7. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
8. Un fichier splicer.properties est créé pour l'application. L'emplacement du fichier splicer.properties est la valeur d'une nouvelle propriété `com.ibm.websphere.xs.sessionFilterProps`. Pour rechercher la propriété personnalisée, accédez à **Serveurs > Types de serveur > Serveurs d'applications WebSphere**.
9. Dans la zone des **serveurs d'applications**, sélectionnez le **nom du serveur**.
10. Dans la zone **Infrastructure du serveur**, sélectionnez **Propriétés personnalisées**.
11. Mettez à jour le fichier `splicer.properties` avec les valeurs que vous avez obtenues dans Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server.
12. Redémarrez les processus serveur d'applications.

Remarque : Changez `splicer.properties` au niveau du gestionnaire de déploiement pour que les propriétés soient synchronisées sur l'agent de noeud. Si vous mettez à jour `splicer.properties` au niveau du noeud, le gestionnaire de déploiement remplace le fichier `splicer.properties` lors de la synchronisation suivante.

Remarque : Si vous revenez dans la gestion de session de base de données et la gestion de session WebSphere eXtreme Scale, le fichier `splicer.properties` est recréé et les modifications que vous avez apportées sont remplacées. Pour plus d'informations sur le processus de synchronisation de fichier depuis le gestionnaire de déploiement vers les notes et connaître les éléments modifiés, voir Synchronisation des fichiers de gestion de système.

Résultats

Vous venez de changer les paramètres de configuration précédents de la gestion de session de mémoire à mémoire ou de base de données avec la gestion de session WebSphere eXtreme Scale.

Rubrique parent : Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

| Suivant >

Tutoriel : Démarrer avec des applications de grille de données simples

Vous pouvez utiliser le modèle d'application de démarrage pour vérifier la connexion entre l'installation de votre client et le dispositif. Cet exemple présente les grilles de données d'entreprise.

Objectifs d'apprentissage

- Procédure de création des grilles de données dans l'interface utilisateur.
- Description du développement d'une application client dans les langages de programmation Java ou .NET. Apprenez à interopérer entre les langages de programmation en créant une grille de données d'entreprise.
- Exécution de l'application client pour insérer des données dans la grille de données.
- Surveillance des grilles de données avec la console Web.

Durée

60 minutes

| Suivant >

Leçon 1.1 du tutoriel d'initialisation : Définition de grilles de données

Vous pouvez définir des grilles de données dans l'interface utilisateur. Pour les besoins de ce tutoriel, créez une grille de données que vous appellerez `my_simple_data_grid`.

Tâches associées:

Création de grilles de données simples

Création de grilles de données simples

Une grille de données simple vous permet d'effectuer des opérations de création, de récupération, de mise à jour et de suppression.

- Le dispositif doit être initialisé et configuré. Pour plus d'informations, voir Installation de WebSphere DataPower XC10 Appliance.

Créez la grille de données simple. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Grille de données simple**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données simple grille de données à créer. Pour les besoins de ce tutoriel, créez une grille de données que vous appellerez `my_simple_data_grid`.

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à :

- Créer une grille de données simple dans l'interface utilisateur.

Module 2 du guide d'initiation : Création d'une application client

Ecrire des applications client pour insérer, mettre à jour, supprimer et extraire des données depuis la grille de données. Vous pouvez utiliser l'exemple d'application pour apprendre à créer une application pour votre environnement.

Objectifs d'apprentissage

A la fin de ce module, vous saurez :

- Java Développer une application client Java
- .NET Développer une application client .NET
- Développer une application de grille de données d'entreprise

Les leçons dans ce module

Leçon 2.1 du tutoriel d'initiation : Création d'une application client Java

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple d'initiation inclut une application client Java que vous pouvez utiliser pour en savoir plus sur la création de votre propre application client.

Leçon 2.2 du tutoriel d'initiation : Création d'une application .NET

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple du tutoriel d'initiation contient une application client .NET que vous pouvez utiliser pour apprendre à créer votre propre application client.

Leçon 2.3 : Création d'une application de grille de données

Pour créer une application de grille de données dans laquelle les clients Java et .NET peuvent mettre à jour la même grille de données, vous devez rendre les classes compatibles. Dans les exemples d'application du tutoriel d'initiation, l'exemple d'application .NET a des alias pour correspondre aux valeurs par défaut Java.

Leçon 2.1 du tutoriel d'initiation : Création d'une application client Java

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple d'initiation inclut une application client Java que vous pouvez utiliser pour en savoir plus sur la création de votre propre application client.

Le fichier `Client.java` dans le répertoire *racine_install_wxs*`/ObjectGrid/gettingstarted/client/src/` est le programme client qui montre comment se connecter à un serveur de catalogue, obtenir l'instance ObjectGrid et utiliser l'API ObjectMap. L'API ObjectMap stocke les données comme paires clé-valeur et elle est idéale pour la mise en cache d'objets qui n'ont aucune relation. Les étapes suivantes présentent le contenu du fichier `Client.java`.

1. Connectez-vous au service de catalogue en obtenant une instance ClientClusterContext.

   Pour établir la connexion au serveur de catalogues, utilisez la méthode connect de l'API ObjectGridManager. Le fragment de code suivant montre comment se connecter à un serveur de catalogue et obtenir une instance ClientClusterContext :

   ```
   ClientClusterContext ccc =
   ObjectGridManagerFactory.getObjectGridManager().connect(cep, null, null);
   ```

   La méthode connect tente de se connecter à chaque appliance de la liste jusqu'à ce qu'elle puisse établir une connexion. Un basculement automatique est effectué si l'un des autres dispositifs ne répond pas.

   Si les connexions aux serveurs de catalogue aboutissent, la méthode connect retourne une instance ClientClusterContext. L'instance ClientClusterContext est nécessaire pour obtenir la grille d'objets ObjectGrid depuis l'API ObjectGridManager.
2. Obtenez une instance ObjectGrid.

   Pour obtenir une instance ObjectGrid, utilisez la méthode getObjectGrid de l'API ObjectGridManager. La méthode getObjectGrid requiert l'instance ClientClusterContext et le nom de l'instance de grille de données. L'instance ClientClusterContext est obtenue pendant la connexion au serveur de catalogue. Le nom de l'instance de grille de données est le nom de la grille de données simple que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir la grille de données en appelant la méthode getObjectGrid de l'API ObjectGridManager.

   ```
   ObjectGrid grid = ObjectGridManagerFactory.getObjectGridManager().getObjectGrid(ccc,
   "my_simple_data_grid");
   ```

3. Définissez les informations d'identification de sécurité nécessaires.

   Créez une configuration de sécurité du client avec un nom d'utilisateur et un mot de passe que vous fournissez à l'application. Le nom d'utilisateur et le mot de passe que vous utilisez doivent avoir l'autorisation d'accéder à la grille de données sur le dispositif. Pour plus d'informations sur la création d'un utilisateur autorisé, voir Gestion des utilisateurs et des groupes.

   ```
                   // Creates a ClientSecurityConfiguration object using the specified
   file
                   ClientSecurityConfiguration clientSC =
   ClientSecurityConfigurationFactory.getClientSecurityConfiguration();
                   clientSC.setSecurityEnabled(true);
                   // Creates a CredentialGenerator using the passed-in user and
   password.
                   CredentialGenerator credGen = new
   UserPasswordCredentialGenerator(username,password);
                   clientSC.setCredentialGenerator(credGen);
                   return clientSC;
   ```

4. Obtenez une instance Session.

   Vous pouvez obtenir une session de l'instance ObjectGrid obtenue. Une instance Session est indispensable pour obtenir l'instance ObjectMap et pour effectuer une démarcation de transaction. Le fragment de code suivant montre comment obtenir une instance Session en appelant la méthode getSession de l'API ObjectGrid.

```
Session sess = grid.getSession();
```

5. Obtenez une instance ObjectMap.

   Après avoir obtenu une instance Session, vous pouvez obtenir une instance ObjectMap depuis une instance Session en appelant la méthode getMap de l'API Session. Le nom d'instance de mappe que vous envoyez à la méthode getMap porte le nom de la grille de données que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir ObjectMap en appelant la méthode getMap de l'API Session.

   ```
   ObjectMap map1 = sess.getMap("my_simple_data_grid");
   ```

   L'exemple précédent utilise l'instance de mappe par défaut qui est nommé d'après la grille de données. Vous pouvez également indiquer un nouveau nom de mappe, comme dans les exemples suivants :

   ```
   ObjectMap map2 = sess.getMap("my_simple_data_grid.CT.P");
   ObjectMap map3 = sess.getMap("my_new_map.NONE");
   ```

   La mappe `my_simple_data_grid.CT.P` est une mappe qui utilise l'expulsion en fonction de l'heure de création et le verrouillage pessimiste. La mappe `my_new_map.NONE` ne dispose pas de paramètres d'expulsion ou de verrouillage. Pour plus d'informations, voir Options de configuration de mappe dynamique.

6. Utilisez les méthodes ObjectMap.

   Une fois une instance ObjectMap obtenue, vous pouvez utiliser l'API ObjectMap. N'oubliez pas que l'interface ObjectMap est une mappe transactionnelle et qu'elle requiert une démarcation de transaction à l'aide des méthodes begin et commit de l'API Session. Faute de démarcation de transaction explicite, les opérations ObjectMap s'exécutent avec des transactions de validation automatique.

   La clé que vous utilisez peut avoir n'importe quel type Java existant, tel que java.lang.String ou Entier. Les valeurs peuvent correspondre à n'importe quel type d'objet sérialisable.

   - Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une transaction de validation automatique.

     ```
     map1.insert(key1, value1);
     ```

   - Vous pouvez exécuter une transaction sur une seule partition à la fois ou sur plusieurs partitions. Pour exécuter une transaction sur une seule partition, utilisez une transaction de validation en une phase :

     ```
     sess.setTxCommitProtocol(TxCommitProtocol.ONEPHASE);
     sess.begin();
     map1.insert(k, v);
     sess.commit();
     ```

     Pour exécuter une transaction sur plusieurs partitions, utilisez une transaction de validation en deux phases :

     ```
     sess.setTxCommitProtocol(TxCommitProtocol.TWOPHASE);
     sess.begin();
     map1.insert(k, v);
     sess.commit();
     ```

7. Facultatif : Fermez la session. Une fois toutes les opérations Session et ObjectMap terminées, fermez la session à l'aide de la méthode Session.close(). Cette méthode renvoie les ressources qui étaient utilisées par la session.

   ```
   sess.close();
   ```

   Par conséquent, les appels suivants de la méthode getSession() sont plus rapides, et moins d'objets Session se trouvent dans le segment.

Concepts associés:

Développement d'applications de grilles de données avec des API Java

Tâches associées:

Accès à la documentation des API Java

Information associée:

Documentation sur les API

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à créer une application client simple pour effectuer des opérations de grille de données.

< Précédent | **Suivant >**

Leçon 2.2 du tutoriel d'initiation : Création d'une application .NET

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple du tutoriel d'initiation contient une application client .NET que vous pouvez utiliser pour apprendre à créer votre propre application client.

- Vous devez avoir installé WebSphere eXtreme Scale Client pour .NET. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client for .NET.
- Le fichier de projet de l'exemple fonctionne avec Microsoft Visual Studio 2010 et les versions suivantes. Si vous utilisez une version précédente de Microsoft Visual Studio, vous devez créer votre propre fichier de projet.

Vous pouvez utiliser l'exemple d'application .NET du tutoriel pour :

- Vérifier que vous avez installé correctement WebSphere eXtreme Scale Client for .NET.
- Apprendre à écrire des applications pour le client .NET qui communiquent avec la grille de données pour créer des applications personnalisées. L'exemple montre comment se connecter à une grille de données sur un serveur de catalogue distant. Le mode interactif montre comment exécuter des transactions manuelles en utilisant la mappe GridMapPessimisticTx. Le mode de ligne de commande montre des transactions validées automatiquement avec la mappe GridMapPessimisticAutoTx.
- Apprendre à interagir avec l'exemple du tutoriel d'initiation iJava™. Les deux exemples d'applications stockent les éléments dans la grille de données avec les paires TestKey/TestValue. L'exemple .NET contient les attributs ClassAlias et FieldAlias pour créer des identificateurs uniques pour la sérialisation et la désérialisation. Si une opération d'insertion de clé est exécutée depuis l'application client Java, le client .NET peut obtenir la valeur en exécutant une opération get sur la clé insérée.

L'exemple d'application du tutoriel d'initiation .NET a les limitations suivantes :

- Seul le verrouillage pessimiste est pris en charge.
- Les opérations de validation en deux phases sont prises en charge. Vous pouvez valider les opérations dans une seule partition. Si vous exécutez une validation qui implique plusieurs partitions, une exception MultiplePartitionWriteException est générée.
- L'exemple ne prend pas en charge les valeurs null. L'API .NET autorise les valeurs null, mais vous devez utiliser des types pouvant avoir la valeur "null".

Le fichier de projet `SimpleClient.csproj` se trouve dans le répertoire *net_client_home*`/sample/SimpleClient`. Ce fichier de projet est le programme client qui montre comment connecter un serveur de catalogue, obtenir l'instance ObjectGrid et utiliser l'API ObjectMap. L'API ObjectMap stocke les données comme paires clé-valeur et elle est idéale pour la mise en cache d'objets qui n'ont aucune relation. Les étapes suivantes contiennent des informations sur le contenu de clé du fichier `SimpleClient.csproj`. Vous pouvez également consulter le fichier de projet plus en détail dans Microsoft Visual Studio.

Le tutoriel montre comment utiliser IGridMapPessimisticTx qui est la mappe de transactions manuelles utilisée lorsque l'application est exécutée en mode interactif. Si vous utilisez l'application en mode de ligne de commande, la mappe IGridMapPessimisticAutoTx est utilisée.

1. Connectez-vous au service de catalogue en obtenant une instance IClientConnectionContext.

   Pour vous connecter au serveur de catalogue, utilisez la méthode Connect de l'API IGridManager.

   ```
   IGridManager gm = GridManagerFactory.GetGridManager( );
   ICatalogDomainInfo cdi = gm.CatalogDomainManager.CreateCatalogDomainInfo( endpoint
   );
   ccc = gm.Connect( cdi, "SimpleClient.properties" );
   ```

   Si la connexion au serveur de catalogue aboutit, la méthode Connect retourne une instance IClientConnectionContext. L'instance IClientConnectionContext est nécessaire pour obtenir la grille de données de l'API IGridManager.

2. Obtenez une instance ObjectGrid.

   Pour obtenir une instance ObjectGrid, utilisez la méthode GetGrid de l'API IGridManager. La méthode GetGrid nécessite l'instance IClientConnectionContext et le nom de l'instance de grille de données. L'instance IClientConnectionContext est obtenue pendant la connexion au serveur de catalogue. Le nom de l'instance de grille de commande est la grille définie dans le fichier `objectgrid.xml`.

   ```
   grid = gm.GetGrid( ccc, gridName );
   ```

3. Obtenez une instance de mappe.

   Vous pouvez obtenir une instance de mappe en appelant la méthode GetGridMapPessimisticTx de l'API IGrid. Envoyez le nom de la mappe comme paramètre à la méthode GetGridMapPessimisticTx pour obtenir l'instance de mappe.

   ```
   pessMap = grid.GetGridMapPessimisticTx<Object, Object>( mapName );
   ```

4. Utilisez les méthodes IGridMapPessimisticTx.

   Une fois une instance de mappe obtenue, vous pouvez utiliser l'API IGridMapPessimisticTx.

   Le fragment de code suivant montre comment utiliser l'API IGridMapPessimisticTx.

   - Pour lancer une transaction avec l'API IGridMapPessimisticTx, vous devez appeler la méthode map.Transaction.Begin(). Cette méthode lance une nouvelle transaction dans laquelle vous pouvez exécuter des opérations.

     ```
     case "begin":
        map.Transaction.Begin( );
        return 0;
     ```

   - La méthode add insère une nouvelle paire clé/valeur. Si la clé existe, une exception est émise.

     ```
      case "a":
        if( key == null ) throw new MissingParameterException( "key" );
        if( value == null ) throw new MissingParameterException( "value" );
        map.Add( key, value );
        Console.WriteLine( "SUCCESS: Added key '{0}' with value '{1}',
        partitionId={2}", key, value, partitionId );
        return 0;
     ```

   - La méthode put insère ou met à jour une paire clé/valeur.

     ```
     case "p":
       if( key == null ) throw new MissingParameterException( "key" );
       if( value == null ) throw new MissingParameterException( "value" );
       map.Put( key, value );
       Console.WriteLine( "SUCCESS: Put key '{0}' with value '{1}',
       partitionId={2}", key, value, partitionId );
       return 0;
     ```

   - La méthode replace remplace une paire clé/valeur existante. Si l'élément n'est pas présent, une exception est émise.

     ```
     case "r":
       if( key == null ) throw new MissingParameterException( "key" );
       if( value == null ) throw new MissingParameterException( "value" );
       map.Replace( key, value );
       Console.WriteLine( "SUCCESS: Replaced key '{0}' with value '{1}',
       partitionId={2}", key, value, partitionId );
       return 0;
     ```

   - La méthode remove supprime une paire clé/valeur.

     ```
     case "d":
       if( key == null ) throw new MissingParameterException( "key" );
       map.Remove( key );
       Console.WriteLine( "SUCCESS: Deleted value with key '{0}',
       partitionId={1}", key, partitionId );
              return 0;
     ```

   - La méthode get extrait la valeur de la clé.

     ```
      case "g":
       if( key == null ) throw new MissingParameterException( "key" );
       value = ( TestValue )map.Get( key );
       if ( value != null )
       {
     ```

```
    Console.WriteLine( "SUCCESS: Value is '{0}',
  partitionId={1}", value, partitionId );
  }
  else
  {
    Console.WriteLine( "FAILED: Key not found" );
  }
  return 0;
```

- Si vous voulez annuler les opérations que vous avez exécutées dans l'opération avant la validation, utilisez la méthode rollback.

```
case "rollback":
  map.Transaction.Rollback( );
  return 0;
```

- La méthode commit valide les opérations exécutées dans la transaction.

```
case "commit":
  map.Transaction.Commit( );
  return 0;
```

Tâches associées:

.NET Configuration de l'environnement de développement .NET
.NET Accès à la documentation des API WebSphere eXtreme Scale Client for .NET

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à créer une application .NET simple pour exécuter des opérations de grille de données.

< Précédent | **Suivant >**

Leçon 2.3 : Création d'une application de grille de données

Pour créer une application de grille de données dans laquelle les clients Java™ et .NET peuvent mettre à jour la même grille de données, vous devez rendre les classes compatibles. Dans les exemples d'application du tutoriel d'initiation, l'exemple d'application .NET a des alias pour correspondre aux valeurs par défaut Java.

Ajoutez des alias de classe et des attributs d'alias de zone à l'application .NET. Vous pouvez ajouter les alias de classe à l'application .NET, l'application Java ou aux deux applications. L'exemple .NET a des alias qui correspondent aux valeurs par défaut Java. Par conséquent, l'application Java n'a pas besoin d'alias. Les fichiers `TestKey.cs` et `TestValue.cs` se trouvent dans le répertoire *net_client_home*`/sample/SimpleClient`.

Figure 1. Attribut d'alias de classe dans le fichier TestKey.cs

```
[ClassAlias( "com.ibm.websphere.xs.sample.gettingstarted.model.TestKey" )]
```

Figure 2. Attribut d'alias de classe dans le fichier TestValue.cs

```
[ClassAlias( "com.ibm.websphere.xs.sample.gettingstarted.model.TestValue" )]
```

Concepts associés:

Annotations ClassAlias et FieldAlias

Tâches associées:

Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Point de contrôle de la leçon

Vous avez ajouté des attributs à l'application d'initiation .NET. Par conséquent, vous pouvez utiliser l'application d'initiation Java en créant une grille de données d'entreprise.

Module 3 : Exécution de l'exemple d'application dans la grille de données

Vous pouvez exécuter l'exemple d'application client sur la grille de données de votre dispositif.

En revanche, la manière d'exécuter cette exemple d'application client varie selon qu'il s'agit de la version Java ou de la version .NET.

Objectifs d'apprentissage

A la fin de ce module, vous saurez :

- Java Exécuter l'exemple d'application client Java de l'initiation
- .NET Exécuter l'exemple d'application client .NET.

Vous pouvez exécuter les exemples d'application Java et .NET séparément, mais vous pouvez également les exécuter simultanément sur la même grille de données. Par exemple, vous pouvez insérer une valeur dans la grille de données avec l'application .NET, puis obtenir cette valeur avec l'application Java. Dans ce scénario, vous exécutez une grille de données d'entreprise.

Les leçons dans ce module

Leçon 3.1 du tutoriel d'initiation : Exécution de l'exemple d'application Java

Procédez comme suit pour exécuter un client Java pour interagir avec la grille de données.

Leçon 3.2 du tutoriel d'initiation : Exécution de l'exemple d'application .NET

Procédez comme suit pour exécuter une application WebSphere eXtreme Scale Client for .NET pour interagir avec la grille de données. Le serveur de catalogue, le serveur de conteneur et le client s'exécutent tous sur un serveur unique dans cet exemple.

< Précédent | **Suivant >**

Leçon 3.1 du tutoriel d'initiation : Exécution de l'exemple d'application Java

Procédez comme suit pour exécuter un client Java pour interagir avec la grille de données.

Modifiez le fichier *racine_install_wxs*`/ObjectGrid/gettingstarted/env.bat|sh`. Ce fichier est automatiquement appelé par le client. Il contient les informations suivantes :

```
SET CATALOGSERVER_HOST=<nom_hôte_xc10>
SET CATALOGSERVER_PORT=2809
SET GRID_NAME=ma_grille_données_simple
SET MAP_NAME=my_map.P
```

Pour la connexion à une grille de données sur le dispositif, vous devez mettre à jour les propriétés **CATALOGSERVER_HOST** et **CATALOGSERVER_PORT**. Le serveur de catalogue à spécifier est affiché dans la page d'interface utilisateur de la grille de données simple que vous avez créée. Cliquez sur **Grille de données** > **Grille de données simple** > ***ma_grille_données_simple*** et utilisez les valeurs dans la zone **Services de catalogue**. La valeur de la propriété **GRID_NAME** doit correspondre au nom de la grille de données que vous avez créée. La propriété **MAP_NAME** crée une mappe sans durée de vie (TTL) et utilisant le verrouillage pessimiste. Pour plus d'informations sur le nommage des mappes dynamiques, voir Options de configuration de mappe dynamique.

- Exécutez le client en mode interactif. Dans la fenêtre de ligne de commande, exécutez l'une des commandes suivantes :
  - UNIX Linux `./runclient.sh`
  - Windows `runclient.bat`

  1. Démarrez une transaction. Vous pouvez utiliser une opération de validation en une phase ou deux phases pour la transaction. Avec la validation en une phase, la transaction doit écrire dans une seule partition. Si pendant la transaction vous insérez des clés dans différentes partitions, la validation de la transaction échoue. Vous pouvez utiliser la validation en deux phases pour écrire dans plusieurs partitions au cours d'une même transaction.
     - Démarrez une transaction avec validation en une phase.

       ```
       begin
       ```

     - Démarrez une transaction avec validation en deux phases.

       ```
       begin2pc
       ```

  2. Insérez une valeur.

     ```
     > i key1 helloWorld
     SUCCESS: Inserted TestValue [value=helloWorld] with key TestKey [key=key1], p
     art
     itionId=6
     ```

  3. Extrayez une valeur que vous avez insérée.

     ```
     > g key1
     Value is TestValue [value=helloWorld], partitionId=6
     ```

  4. Mettez à jour une valeur.

     ```
     > u key1 goodbyeWorld
     SUCCESS: Updated key TestKey [key=key1] with value TestValue [value=goodbyeWo
     rld
     ], partitionId=6
     ```

  5. Annulez la transaction. Lorsque vous annulez la transaction, toutes les opérations associées à la transaction sont annulées.

     ```
     > rollback
     ```

6. Pour tester l'opération d'annulation, essayez d'obtenir de nouveau la clé. Comme vous avez annulé la transaction, la clé n'existe pas :

```
> g key1
```

7. Insérez une valeur.

```
> i key1 helloWorld
SUCCESS: Inserted TestValue [value=helloWorld] with key TestKey [key=key1], p
art
itionId=6
```

8. Validez la valeur. Après avoir validé la transaction, vous ne pouvez pas annuler les modifications.

```
> commit
```

9. Supprimez une valeur que vous avez insérée.

```
> d key1
SUCCESS: Deleted value with key TestKey [key=key1], partitionId=6
```

10. Insérez des entrées de test. Par exemple, pour insérer 1000 clés et valeurs numérotées de 0 à 999, utilisez la commande suivante :

```
> n 1000
```

- Exécutez le client en mode de ligne de commande. Le mode de ligne de commande peut être utile si vous voulez écrire un script pour exécuter l'application client. Vous pouvez exécuter les mêmes commandes que celles que vous exécutez en mode interactif. Voici un exemple de syntaxe pour le mode de ligne de commande :
  - UNIX Linux

```
./runclient.sh i "key1" "helloWorld"
```

  - Windows

```
runclient.bat i "key1" "helloWorld"
```

Point de contrôle de la leçon

Points étudiés

Dans cette leçon, vous avez appris à :

- Exécuter l'exemple d'application client Java pour insérer, obtenir, mettre à jour et supprimer des données de la grille de données.

< Précédent | **Suivant >**

Leçon 3.2 du tutoriel d'initiation : Exécution de l'exemple d'application .NET

Procédez comme suit pour exécuter une application WebSphere eXtreme Scale Client for .NET pour interagir avec la grille de données. Le serveur de catalogue, le serveur de conteneur et le client s'exécutent tous sur un serveur unique dans cet exemple.

WebSphere eXtreme Scale Client for .NET ne prend en charge les validations en une seule phase. Par conséquent, si vous tentez d'insérer plusieurs valeurs dans une même transaction, une exception risque de se produire, car les valeurs sont placées dans des partitions différentes. Pour empêcher l'occurrence de ces exceptions lorsque vous exécutez l'exemple, vous pouvez changer le fichier XML descripteur de règle de déploiement pour utiliser une seule partition. Pour plus d'informations sur la mise à jour du nombre de partitions, voir Leçon 1.1 du tutoriel d'initialisation : Définition de grilles de données.

Vous pouvez exécuter l'exemple d'application en mode interactif ou de ligne de commande. En mode interactif, l'application exécute les transactions de grille de données manuelles avec l'API IGridMapPessimisticTx. Le mode de ligne de commande exécute les transactions de grille de données automatiques avec l'API IGridMapPessimisticAutoTx.

Vous pouvez exécuter l'exemple en mode interactif ou de ligne de commande :

- Exécutez l'exemple d'application client en mode interactif.
  1. Exécutez l'application client simple. Le fichier se trouve dans le répertoire *net_client_home*\gettingstarted\bin\. Pour exécuter l'exemple en mode interactif, exécutez la commande suivante :

     ```
     SimpleClient.exe -i [-h <nom_hôte:port>]  [-g <nom_grille>] [-m <nom_mappe>]
     ```

     -h <nom_hôte:port>

     Indique le nom d'hôte et le port du serveur de catalogue auquel vous voulez vous connecter. Le serveur de catalogue à spécifier est affiché dans la page d'interface utilisateur de la grille de données simple que vous avez créée. Cliquez sur **Grille de données** > **Grille de données simple** > ***ma_grille_données_simple*** et utilisez les valeurs dans la zone **Services de catalogue**.

     -g <nom_grille>

     Indique le nom de la grille de données à utiliser. Vous devez utiliser une mappe avec stratégie de verrouillage pessimiste. Pour ce tutoriel, utilisez `ma_grille_données_simple` ou le nom que vous avez indiqué lorsque vous avez créé la grille de données dans le dispositif. Si vous n'indiquez pas de nom, la grille de données `Grid` est utilisée.

     -m <nom_mappe>

     Indique le nom de la mappe à utiliser. Pour ce tutoriel, indiquez `ma_mappe.NONE.P`. Si vous n'indiquez pas de valeur, une erreur se produit. Pour plus d'informations concernant les noms de mappe, voir Options de configuration de mappe dynamique.

     Si vous exécutez l'application sans paramètre, l'aide de l'application s'affiche.

  2. Affichez la liste des commandes disponibles.

     ```
     Enter a command: help
     This program executes simple CRUD operations on a map.
            a - Adds a value with the specified key. If the key already exists,
            DuplicateKeyException is thrown
            p - Adds a value with the specified key, replacing the entry if it
            already exists
            r - Replaces the value of the specified key. If the key does not exis
     t,
            a CacheKeyNotFound exception is thrown
            g - Retrieve and display the value of the specified key
            d - Deletes the key
            gp - Gets the partition id for the key
            ck - Checks if the map contains the key
            h - Display help
            begin - Begin manual transaction
            commit - Commit transactions
     ```

```
        rollback - Rollback transactions
        exit - Exit program
```

3. Démarrez la transaction. Vous devez démarrer une transaction pour pouvoir exécuter des commandes sur la grille de données. Si vous ne démarrez pas la transaction, une exception NoActiveTransacationException se produit.

```
Enter a command: begin
```

4. Ajoutez des données à la grille.

```
Enter a command: a key1 value1
SUCCESS: Added 'TestKey [key=key1]' with value 'TestValue [value=value1]',
partitionId=6
```

5. Recherchez et affichez la valeur.

```
Enter a command: g key1
SUCCESS: Value is 'TestValue [value=value1]', partitionId=6
```

   Dans cet exemple, `value1` est retourné.

6. Mettez à jour la clé. Utilisez la commande put qui ajoute une valeur avec la clé définie en remplaçant la valeur existante éventuelle.

```
Enter a command: p key1 value2
SUCCESS: Put key 'TestKey [key=key1]' with value 'TestValue [value=value2]',
 partitionId=6
Enter a command: g key1
SUCCESS: Value is 'TestValue [value=value2]', partitionId=6
```

7. Remplacez la clé. La commande remplace la valeur par la clé définie. Si la clé n'existe pas, une exception CacheKeyException est émise.

```
Enter a command: r key1 value3
SUCCESS: Replaced key 'TestKey [key=key1]' with value 'TestValue [value=value
3]'
, partitionId=6
```

8. Annulez la transaction et essayez d'afficher de nouveau la clé de valeur. Vous pouvez annuler la transaction à tout moment avant la validation.

```
Enter a command: rollback
Enter a command: begin
Enter a command: g key1
FAILED: Key not found
```

   Lorsque vous exécutez la commande **get**, vous obtenez une erreur signalant que la clé est introuvable.

9. Validez une clé et une valeur dans la grille de données.

```
Enter a command: begin
Enter a command: a key2 value2
SUCCESS: Added 'TestKey [key=key2]' with value 'TestValue [value=value2]',
partitionId=7
Enter a command: commit
```

10. Obtenez un ID de partition pour une clé.

```
Enter a command: begin
Enter a command: gp key2
SUCCESS: partitionId=7
```

11. Recherchez des clés dans la mappe.

```
Enter a command: ck key2
```

```
SUCCESS: The map contains key 'TestKey [key=key2]'
Enter a command: ck key3
SUCCESS: The map does NOT contain key 'TestKey [key=key3]'
```

12. Supprimez la clé et quittez.

```
Enter a command: begin
Enter a command: d key2
SUCCESS: Deleted value with key 'TestKey [key=key2]', partitionId=7
Enter a command: commit
Enter a command: exit
```

- Exécutez le client en mode de ligne de commande. Le mode de ligne de commande exécute les transactions de grille de données automatiques avec l'API IGridMapPessimisticAutoTx. Pour utiliser ce mode, transmettez l'action sur la ligne de commande. Le mode de ligne de commande peut être utile si vous voulez écrire un script pour exécuter l'application client. Vous pouvez exécuter les mêmes commandes que celles que vous exécutez en mode interactif. Voici un exemple de syntaxe pour le mode de ligne de commande :

```
SimpleClient [-h <hôte:port>] [-g <nom_grille>] [-m <nom_mappe>] <a | p | r | g |
d> <clé> [<valeur>]
```

Tâches associées:

.NET Développement d'applications de grille de données avec les API .NET
Accès à la documentation des API WebSphere eXtreme Scale Client for .NET

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à :

- Exécuter l'exemple d'application .NET pour insérer, obtenir, mettre à jour et supprimer des objets de la grille de données.

< Précédent | **Suivant >**

< Précédent

Leçon 4 du tutoriel du guide de démarrage : Surveillance de l'environnement

Vous pouvez utiliser l'utilitaire **xscmd** et les outils de la console Web pour surveiller votre environnement de grille de données.

Tâches associées:

Surveillance des grilles de données dans l'interface utilisateur

Surveillance des grilles de données dans l'interface utilisateur

L'interface utilisateur vous permet de visualiser les performances globales des grilles des données présentes dans votre environnement. Sa section dédiée à la surveillance offre une vue globale de toutes les grilles de données du dispositif, une vue globale de chaque grille, et des rapports détaillés sur les grilles individuelles.

Pour plus d'informations concernant la surveillance des grilles de données dans l'interface utilisateur, voir Surveillance des grilles de données dans l'interface utilisateur.

Surveillance avec l'utilitaire `xscmd`

1. Facultatif : Si l'authentification de client est activée : Sur l'installation client, ouvrez une fenêtre de ligne de commande. Sur la ligne de commande, définissez les variables d'environnement appropriées.
2. Accédez au répertoire *rép_base_wxs*`/bin`.

   `cd` *rép_base_wxs*`/bin`

3. Plusieurs commandes vous permettent d'afficher des informations concernant votre environnement.
   - Afficher tous les serveurs de conteneur en ligne pour la grille de données de la grille et le groupe de mappes mapSet :

     ```
     xscmd -c showPlacement -g Grid -ms mapSet
     ```

   - Afficher les informations de routage de la grille de données :

     ```
     xscmd -c routetable -g Grid
     ```

   - Afficher le nombre d'entrées de mappe dans la grille de données :

     ```
     xscmd -c showMapSizes -g Grid -ms mapSet
     ```

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à :

- surveiller les statistiques de la grille et des serveurs ;

< Précédent

Planification de l'environnement DataPower XC10 Appliance

Pour pouvoir intégrer DataPower XC10 Appliance à la topologie, l'environnement doit répondre aux conditions et à la configuration logicielle suivantes.

Configuration requise
Votre environnement doit présenter la configuration suivante pour permettre l'intégration avec WebSphere DataPower XC10 Appliance.

Conventions relatives aux répertoires
Les conventions de répertoire suivantes sont utilisées dans toute la documentation pour faire référence à des répertoires spéciaux, tels que *wxs_install_root* et *wxs_home*. Vous pouvez accéder à ces répertoires pendant plusieurs scénarios différents, y compris lors de l'installation et de l'utilisation des outils de ligne de commande.

Ports réseau
Si vous utilisez WebSphere DataPower XC10 Appliance derrière un pare-feu, vous devez activer la communication via les ports ci-dessous.

Conditions nécessaires à l'installation d'IBM WebSphere DataPower XC10 Appliance
Vous devez satisfaire les exigences en termes de matériel, logiciel, armoire et outils afin d'installer et de configurer IBM® WebSphere DataPower XC10 Appliance. Utilisez cette liste de prérequis pour planifier l'installation, la configuration et l'utilisation d'IBM WebSphere DataPower XC10 Appliance.

Spécifications et fonctions du dispositif
Le matériel de type 7199-92x prend en charge WebSphere DataPower XC10 Appliance version 2.5. Le matériel de type 9235-92x était livré avec la version 1.0 et n'est pris en charge que jusqu'à la version 2.1.

Interopérabilité du produit
WebSphere DataPower XC10 Appliance a été testé du point de vue de l'interopérabilité avec d'autres produits IBM.

.NET 2.5+ ### Remarques relatives à Microsoft .NET
Deux environnements .NET existent dans WebSphere eXtreme Scale : l'environnement de développement et l'environnement d'exécution. Ces environnements ont des exigences spécifiques.

Configuration requise

Votre environnement doit présenter la configuration suivante pour permettre l'intégration avec WebSphere DataPower XC10 Appliance.

Configuration matérielle

Pour effectuer la configuration initiale, vous devez utiliser une connexion série. La connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif WebSphere DataPower XC10 Appliance. Pour établir la connexion série, utilisez le câble série faux modem RJ45/DB-9 fourni.

Remarque : Le terminal n'est pas équipé d'un port série ; utilisez un câble adaptateur USB/Série.

Vous pouvez effectuer une installation automatisée ou à distance via un serveur de terminal connecté au réseau.

Configuration logicielle requise

Le dispositif WebSphere DataPower XC10 Appliance est fourni avec un logiciel IBM. Vous ne pouvez pas installer d'autre logiciel sur le dispositif.

Navigateurs Web requis

L'interface utilisateur prend en charge les navigateurs Web suivants :

- Mozilla Firefox, version 3.5.x et versions ultérieures
- Microsoft Internet Explorer version 7 ou ultérieure

Logiciels pris en charge

Vous pouvez utiliser les logiciels suivants avec WebSphere DataPower XC10 Appliance :

- WebSphere eXtreme Scale Client 8.5
- WebSphere eXtreme Scale Client version 8.6
- WebSphere Application Server version 6.1.0.41 ou ultérieure
- WebSphere Application Server version 7.0.0.21 ou ultérieure
- WebSphere Application Server version 8.0.0.2 ou ultérieure
- WebSphere Application Server version 8.5.0.0 ou ultérieure
- WebSphere Application Server version 8.5.5.0

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Conventions relatives aux répertoires

Les conventions de répertoire suivantes sont utilisées dans toute la documentation pour faire référence à des répertoires spéciaux, tels que *wxs_install_root* et *wxs_home*. Vous pouvez accéder à ces répertoires pendant plusieurs scénarios différents, y compris lors de l'installation et de l'utilisation des outils de ligne de commande.

racine_install_wxs

Le répertoire *wxs_install_root* est le répertoire racine où sont installés les fichiers du produit WebSphere eXtreme Scale. Le répertoire *wxs_install_root* peut être le répertoire dans lequel l'archive d'évaluation est extraite ou à partir duquel le produit est installé WebSphere eXtreme Scale.

- Exemple où la version d'essai a été extraite :

  Exemple : `/opt/IBM/WebSphere/eXtremeScale`

- Exemple lorsque WebSphere eXtreme Scale est installé dans un répertoire autonome :

  UNIX **Exemple :** `/opt/IBM/eXtremeScale`

  Windows **Exemple :** `C:\Program Files\IBM\WebSphere\eXtremeScale`

- Exemple lorsque WebSphere eXtreme Scale est intégré à WebSphere Application Server :

  Exemple : `/opt/IBM/WebSphere/AppServer`

wxs_home

Le répertoire *wxs_home* est le répertoire racine du produit, des bibliothèques, des exemples et des composants WebSphere eXtreme Scale. Ce répertoire est identique au répertoire *wxs_install_root* lorsque l'archive d'évaluation est extraite. Pour les installations autonomes, le répertoire *wxs_home* est le sous-répertoire `ObjectGrid` du répertoire *wxs_install_root*. Pour les installations qui sont intégrées à WebSphere Application Server, ce répertoire est le répertoire `optionalLibraries/ObjectGrid` du répertoire *wxs_install_root*.

- Exemple lorsque la version d'essai a été extraite :

  Exemple : `/opt/IBM/WebSphere/eXtremeScale`

- Exemple lorsque WebSphere eXtreme Scale est installé dans un répertoire autonome :

  UNIX **Exemple :** `/opt/IBM/eXtremeScale/ObjectGrid`

  Windows **Exemple :** *racine_install_wxs*`\ObjectGrid`

- Exemple lorsque WebSphere eXtreme Scale est intégré à WebSphere Application Server :

  Exemple : `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid`

was_root

Le répertoire *was_root* est le répertoire racine d'une installation WebSphere Application Server :

Exemple : `/opt/IBM/WebSphere/AppServer`

.NET **net_client_home**

Le répertoire *net_client* est le répertoire racine d'une installation client .NET.

Exemple : `C:\Program Files\IBM\WebSphere\eXtreme Scale .NET Client`

java_home

Le répertoire *java_home* est le répertoire racine d'une installation de Java™ Runtime Environment Kit (JRE).

UNIX **Exemple :** `/opt/IBM/WebSphere/eXtremeScale/java`

Windows **Exemple :** *racine_install_wxs*`\java`

rep_base_exemples

rep_base_exemples est le répertoire dans lequel vous extrayez les exemples de fichiers qui sont utilisés pour les tutoriels.

UNIX **Exemple :** *rép_base_wxs*`/samples`

Windows **Exemple :** *rép_base_wxs*`\samples`

dvd_root

dvd_root est le répertoire racine du DVD qui contient le produit.

Exemple : `dvd_root/docs/`

rep_base_utilisateur

Le répertoire *rep_base_utilisateur* est l'emplacement de stockage des fichiers utilisateur, tels que les profils de sécurité.

Windows `c:\Documents and Settings\`*user_name*

UNIX `/home/`*user_name*

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Ports réseau

Si vous utilisez WebSphere DataPower XC10 Appliance derrière un pare-feu, vous devez activer la communication via les ports ci-dessous.

Serveur de catalogue

Les ports ci-dessous sont utilisés par le serveur de catalogue. Chacun des trois premiers dispositifs ajoutés à la collectivité exécute un serveur de catalogue.
Port homologue : Utilisé pour la communication entre les serveurs de catalogue. Ce port est défini pour utiliser le numéro 6601.
Port client : Utilisé pour la communication entre les serveurs de catalogue. Ce port est défini pour utiliser le numéro 6602.
Port de service JMX : Utilisé pour les connexions JMX non SSL, notamment pour l'utilitaire xscmd. Ce port est défini pour utiliser le numéro 1099.
Port de connecteur JMX : Utilisé pour les connexions JMX, notamment pour l'utilitaire xscmd. Ce port est défini pour utiliser le numéro 1100 et sélectionne un port entre 7100 et 7116.
Port d'écoute ORB : Utilisé pour la communication entre le client et les serveurs de grille de données. Ce port est défini pour utiliser le numéro 2809.
Port d'écoute CSIv2 : Utilisé pour la communication sécurisée entre le client et les serveurs de grille de données. Ce port est défini pour utiliser le numéro 7499.
Port SNMP : Utilisé pour la surveillance SNMP. Ce port est défini pour utiliser le numéro 161.

Remarque : WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Les ports suivants sont nécessaires à la communication entre les serveurs de conteneur :

Ports de groupe central DCS : Utilisés pour la communication DCS interne de WebSphere DataPower XC10 Appliance. Ce port utilise un port compris entre 6700 et 6716.
Ports d'écoute ORB : Utilisés pour la communication entre le client et les serveurs de grille de données. Ce port utilise un port compris entre 6800 et 6816.
Port d'écoute CSIv2 : Utilisé pour les communications sécurisées entre le client et les serveurs de grille de données. Ce port utilise un port compris entre 7500 et 7516.

Remarque : Si les dispositifs appartiennent à une collectivité et ne sont pas tous situés du même côté du pare-feu, alors les ports DCS, ORB et CSIv2 (serveurs de conteneur), ainsi que les ports 6601 et 6602 (serveurs de catalogue) doivent être ouverts aux communications bidirectionnelles.

Ports de navigateur

Ouvrez les ports 80 et 443 si vous voulez utiliser la console du dispositif ou des services REST. Certaines options de configuration peuvent échouer si ces ports ne sont pas ouverts sur tous les dispositifs de la collectivité.

Serveurs client

WebSphere eXtreme Scale Client utilise un port d'écoute. Ce port est défini dans le fichier de propriétés du client, dans le répertoire *wxs_client_root*\properties.

Application d'identification et de résolution des problèmes

Ouvrez le port 9060 pour l'application d'identification et de résolution des problèmes.

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Conditions nécessaires à l'installation d'IBM WebSphere DataPower XC10 Appliance

Vous devez satisfaire les exigences en termes de matériel, logiciel, armoire et outils afin d'installer et de configurer IBM® WebSphere DataPower XC10 Appliance. Utilisez cette liste de prérequis pour planifier l'installation, la configuration et l'utilisation d'IBM WebSphere DataPower XC10 Appliance.

Compétences requises

Pour configurer et gérer le dispositif, vous devez disposer de compétences en matière d'administration de réseau.

Informations requises

Rassemblez les informations suivantes pour définir la configuration de base de votre système DataPower XC10 Appliance.

- Utilisation de ports Ethernet 1 gigabits ou utilisation de ports Ethernet 10 gigabits pour votre grille de données. Vous devez utiliser tous les ports 1 gigabits ou tous les ports 10 gigabits. Connectez le port de gestion à `MGMT0`.
- Adresse IP et masque de sous-réseau de l'interface Ethernet pour accès à la gestion du dispositif (MGMT).
- Adresse IP des passerelles par défaut (routeurs) prenant en charge les sous-réseaux des interfaces Ethernet.
- Adresse IP du serveur DNS (Domain Name System). Configurez le serveur DNS pour recherche directe et inversée.
- Paramètres de communication pour interface série : 9600.8.n.1 (9600 bauds, 8 bits de données, pas de parité, 1 bit d'arrêt).
- Informations sur le serveur de messagerie (pour configuration des notifications par courrier électronique).
- (Facultatif) Adresses IP et masque de sous-réseau des interfaces Ethernet pour les accès de service au dispositif (ETH0, ETH1 et ETH2).

Exigences relatives à l'armoire

Pour installer le système DataPower XC10 Appliance, vous devez disposer d'une armoire standard de 19 pouces (48,26 cm) avec une profondeur minimale de 25 pouces (63,50 cm) répondant aux critères suivants :

- Présence de colonnes de montage à l'arrière
- Prise en charge de montage avant et arrière

- Présence de colonnes de montage à l'arrière
- Prise en charge de montage avant et arrière

Veillez à ce que l'espace libre requis suivant soit disponible dans l'armoire et autour de celle-ci :

- Au moins 30 pouces (76,20 cm) d'espace libre derrière l'armoire
- Au moins 2 pouces (5,1 cm) au dessus et en dessous du dispositif
- Espace libre suffisant à l'avant pour câbles Ethernet et de console série

La température ambiante dans l'environnement où doit être installé le dispositif ne doit pas dépasser 104° F (40° C).

Outils et équipement

Pour installer le système DataPower XC10 Appliance, vous devez rassembler l'équipement suivant :

- Un tournevis de diamètre moyen
- Deux (2) vis pour armoire standard (fournies avec le dispositif)
- Un à quatre (1-4) câbles réseau
- Câble série ou câble USB/série PL-2303

  Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de pouvoir continuer.

  Remarque : Ne vous débarrassez pas des câbles après l'installation du dispositif. Vous en aurez peut-être besoin pour identifier des problèmes ou effectuer certaines opérations de maintenance à l'avenir.

- Console série avec un connecteur mâle DB9. Il peut s'agir d'un matériel dédié, par exemple une console VT100, ou d'un PC exécutant un émulateur comme HyperTerminal ou Minicom.

Exigences de l'utilisateur interface

Pour utiliser l'interface utilisateur Web, vous pouvez utiliser l'un des navigateurs suivants :

- Mozilla Firefox, version 3.5.x et versions ultérieures

- Microsoft Internet Explorer version 7 ou ultérieure

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Spécifications et fonctions du dispositif

Le matériel de type 7199-92x prend en charge WebSphere DataPower XC10 Appliance version 2.5. Le matériel de type 9235-92x était livré avec la version 1.0 et n'est pris en charge que jusqu'à la version 2.1.

Pour déterminer le type de dispositif, cliquez sur **Dispositif** > **Paramètres** > **Microprogramme**. Le panneau affiche le type de modèle et le numéro de série du dispositif.

Spécifications et fonctions du dispositif type 7199-92x
Utilisez les spécifications et les fonctions pour déterminer l'environnement physique nécessaire pour contenir votre dispositif.

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Spécifications et fonctions du dispositif type 7199-92x

Utilisez les spécifications et les fonctions pour déterminer l'environnement physique nécessaire pour contenir votre dispositif.

A propos des dispositifs de type 7199-92X

Le type 7199-92X WebSphere DataPower XC10 Appliance est livré avec la version 2.0 du logiciel WebSphere DataPower XC10 Appliance. Cette nouvelle version du matériel inclut des processeurs plus rapides, davantage de ports réseau et davantage de capacité de mémoire cache que les versions antérieures.

Pour déterminer le type de dispositif, cliquez sur **Dispositif** > **Paramètres** > **Microprogramme**. Le panneau affiche le type de modèle et le numéro de série du dispositif.

Spécifications

Tableau 1. Spécifications du dispositif de type 7199-92x. Récapitule les spécifications du châssis de type 7199.

Dimensions :	
	7199
Hauteur	8,89 cm (3,5 pouces)
Largeur	42,8 cm (17,25 pouces)
Profondeur	58,4 cm (23 pouces)
Poids	Maximum : 21 kg
Alimentation électrique :	
Tension sinusoïdale	50 - 60 Hz (monophasé) requis
110 Volts CA	Minimum : 100 V_{RMS} Maximum : 127 V_{RMS}
220 Volts CA	Minimum : 200 V_{RMS} Maximum : 240 V_{RMS}
Consommation	10 A pour 110 V CA, 5 A pour 220 V CA Le dispositif de type 7199 contient deux modules de 720 watts. Les deux modules d'alimentation doivent être connectés à la même source de courant pour éviter une différence dans le voltage "terre" entre les deux modules.
Environnement :	
Température ambiante	En opération : • Altitude : de 0 à 914,4 m (3000 ft.)50° à 95° F (10° à 35° C) • Altitude : de 914,4 m (3000 ft.) à 2133,6 m (7000 ft.)50° à 89,6° F (10° à 32° C) Altitude maximale : 2133,6 m (7000 ft.) Hors tension : 50° à 109,4° F (10° à 43° C) Transport : -40° à 140° F (-40° à 60° C)
Humidité	8% à 80%

Caractéristiques

Tableau 2. Options de stockage des données

Caractéristique	Description
Capacité locale	16 Go de stockage sur le système de fichiers local
Grappe de disques durs	Deux unités de disque dur SAS (Serial Attached SCSI) permutables simples de 1 To Capacité : 2 To
Grille de données	Unité Fusion DUO de 320 Go, qui inclut la mémoire cache de 240 Go

Vue avant du type 7199-92x

La vue avant montre les contrôles, les voyants et les connecteurs du dispositif de type 7199. Les modules Ethernet et les modules d'unité de disque dur peuvent être installés à partir du panneau frontal du dispositif de type 7199-92x.

Vue arrière du type 7199-92x

La vue arrière montre les composants et les voyants situés à l'arrière du dispositif. Les modules de ventilation et les modules d'alimentation sont installés depuis l'arrière du dispositif.

Configuration du réseau Ethernet

Les modules Ethernet étendent les options de connectivité réseau. Chaque dispositif a deux modules Ethernet. Les modules Ethernet sont numérotés de gauche à droite, mais si un module a moins de huit ports, il utilisera le numéro de port le plus petit de la plage.

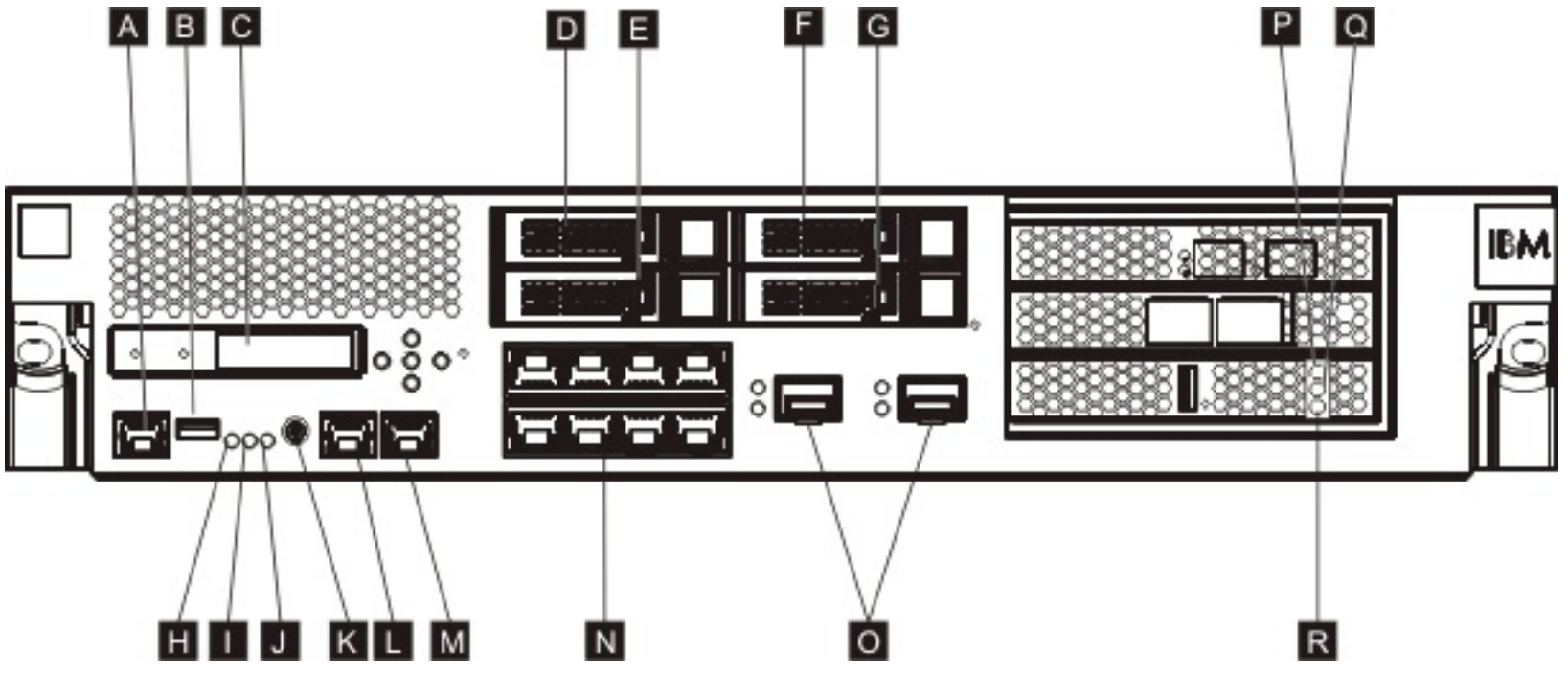
Rubrique parent : Spécifications et fonctions du dispositif

Vue avant du type 7199-92x

La vue avant montre les contrôles, les voyants et les connecteurs du dispositif de type 7199. Les modules Ethernet et les modules d'unité de disque dur peuvent être installés à partir du panneau frontal du dispositif de type 7199-92x.

Diagramme de la vue avant

Figure 1. Vue avant du type 7199-92x



Les libellés du diagramme correspondent aux composants suivants du panneau avant d'un dispositif de type 7199-92x :

A

Connecteur de la console

B

Port USB

C

Module LCM

D

Module d'unité de disque dur 2

E

Module d'unité de disque dur 0

F

Module d'unité de disque dur 3

G

Module d'unité de disque dur 1

H

Voyant d'erreur

I

Voyant d'emplacement

J

Voyant d'alimentation

K

Bouton d'alimentation

L

Connecteur Ethernet `MGT0`

M

Connecteur Ethernet `MGT1`

N

Modules Ethernet de gauche :

- eth0
- eth1

- eth2
- eth3
- eth4
- eth5
- eth6
- eth7

O

Modules Ethernet de droite :

- eth8
- eth9

P

Voyant ambre ou d'erreur du cache.

Q

Voyant jaune ou d'écriture du cache.

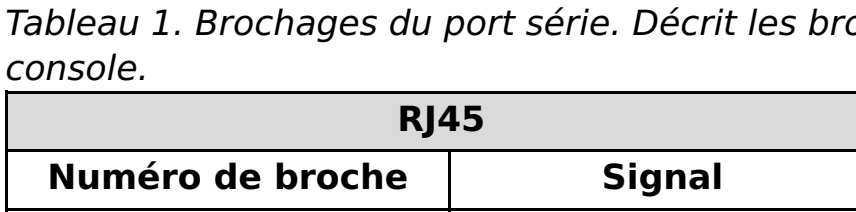
R

Voyant vert ou de lecture du cache.

Module d'affichage

Le panneau avant du dispositif a un module d'affichage à cristaux liquides (LCD) incluant un écran d'affichage LCD et cinq boutons de menu. L'écran LCD fournit des informations sur le type de modèle du dispositif ; cependant, les boutons de menu ne sont pas fonctionnels.

Figure 2. Module d'affichage



Connecteur de la console

Le panneau frontal du dispositif de type 7199 a un connecteur de console. Pour la configuration initiale, utilisez le câble série faux modem RJ45 (IS0 8877) à DB-9 (également appelé DE-9 ou EIA/TIA-562) qui est livré avec le dispositif pour vous connecter à partir d'un terminal ASCII[1] ou pour vous connecter à l'appareil à partir d'un PC utilisant un logiciel d'émulation de terminal. Il y a une connexion RJ45 à une extrémité du câble et une connexion DB-9 série faux modem à l'autre extrémité. Le connecteur RJ45 se connecte au dispositif et le connecteur DB-9 série faux modem se connecte à votre terminal ASCII ou à votre ordinateur personnel. Utilisez le câble adaptateur USB/Série pour connecter le câble à votre ordinateur personnel.

Remarque : Pour la configuration initiale, vous pouvez utiliser le câble de connexion RJ45-série qui est livré avec le dispositif ou bien vous pouvez créer un câble sur la base des spécifications des brochages figurant dans le tableau suivant. N'utilisez pas un câble Ethernet pour connecter le port de la console série à un réseau Ethernet.

Tableau 1. Brochages du port série. Décrit les brochages du port série pour le connecteur de console.

RJ45		DB9	
Numéro de broche	**Signal**	**Numéro de broche**	**Signal**
1	RTS	8	CTS
2	DTR	6	DSR
3	TXD	2	RXD
4	GND	5	GND
5	GND	5	GND
6	RXD	3	TXD
7	DSR	4	DTR
8	CTS	7	RTS

Port USB

Le panneau avant du dispositif a une interface USB conforme aux périphériques USB 2.0. Ce connecteur USB n'est pas activé et ne fournit donc aucune connexion.

Voyants

Le panneau frontal du dispositif de type 7199 a trois voyants autonomes.

Voyant d'erreur

Le voyant d'erreur ambre est allumé si un événement critique est détecté.

Voyant d'emplacement

Le voyant d'emplacement bleu est allumé s'il est activé par le microprogramme. Vous pouvez contrôler si ce voyant est allumé à partir de la ligne de commande. Le voyant reste allumé jusqu'à sa désactivation. Utilisez la commande **`locate-led`** dans l'interface de ligne de commande :

- Pour activer, entrez la commande suivante :

```
locate-led on
```

- Pour désactiver, entrez la commande suivante :

```
locate-led off
```

Voyant d'alimentation

Le voyant d'alimentation est allumé lorsque le dispositif est connecté à une source d'alimentation électrique et que vous avez allumé le dispositif.

- Le voyant d'alimentation vert est allumé lorsque le dispositif est allumé et qu'il est en fonctionnement.
- Si le voyant n'est pas allumé, cela signifie que le dispositif a été éteint.

Bouton d'alimentation

Le bouton d'alimentation se trouve sur le panneau frontal du dispositif. Appuyez sur le bouton pour :

- Allumer le dispositif.
- Commencer un arrêt en douceur (si le dispositif est déjà allumé).

Le fait d'appuyer sur le bouton et de le maintenir enfoncé pendant 5 secondes effectue un arrêt matériel immédiat.

Remarque : Lorsque vous appuyez sur le bouton d'alimentation pour éteindre le dispositif, du courant électrique continue d'y arriver. Pour couper complètement l'électricité au dispositif, déconnectez tous les câbles d'alimentation.

Connecteurs réseau

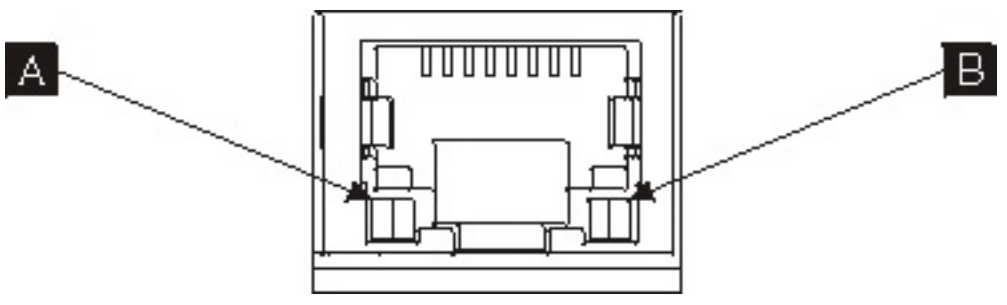
Le panneau frontal de chaque dispositif a deux ports Ethernet de gestion de réseau local et deux modules Ethernet. Voir Configuration du réseau Ethernet pour une description de la convention d'attribution de nom d'Ethernet.

Ports Ethernet de gestion de réseau local

Les deux ports Ethernet de gestion du système permettent la connexion au réseau local. Ces ports offrent un accès pour la gestion à distance de l'unité et ne peuvent pas être utilisés comme ports de données. Les interfaces Ethernet restantes peuvent gérer le trafic de données et les fonctions de connexion de et vers les différents services DataPower.

Méthode recommandée : Utilisez l'interface Ethernet `MGT0` ou `MGT1` pour les fonctions de gestion à l'échelle du système pour gérer le trafic réseau des fonctions entrantes SNMP, SSH et de l'interface utilisateur sur votre intranet.

Figure 3. Voyants des ports Ethernet



Connecteur Ethernet `MGT0`

Cette interface Ethernet peut gérer toutes les données de transactions sur le dispositif. Le connecteur Ethernet `MGT0` prend également en charge IPMI sur réseau local, y compris la liaison série sur réseau local. `MGT0` a un voyant de vitesse et un voyant d'activité associés :

Voyant de vitesse (A)

- Le voyant vert indique une connexion à 1 gigabits par seconde.

- Le voyant ambre indique une connexion à 10 ou à 100 mégabits par seconde.

Voyant d'activité (B)

- Le voyant vert indique que le port est lié.
- Le voyant vert clignotant indique que le port est actif.

Connecteur Ethernet MGT1

Cette interface Ethernet peut gérer toutes les données de transactions sur le dispositif. MGT1 a un voyant d'activité et un voyant de vitesse associés :

Voyant de vitesse (A)

- Le voyant vert indique une connexion à 1 gigabits par seconde.
- Le voyant ambre indique une connexion à 10 ou à 100 mégabits par seconde.

Voyant d'activité (B)

- Le voyant vert indique que le port est lié.
- Le voyant vert clignotant indique que le port est actif.

Modules Ethernet

Le dispositif DataPower a deux modules Ethernet pour la connectivité Ethernet. Le module Ethernet de gauche a huit ports RJ45 et le module Ethernet de droite a deux ports SFP+ (small-form factor pluggable) à 10 gigabits. Le nom de l'interface Ethernet dépend de la configuration du module, avec les noms des interfaces Ethernet dépendant de la configuration du module Ethernet.

Le module 1 gigabits prend en charge Ethernet avec du câble à paire torsadée non protégé, avec des interfaces standard, incluant :

- 10BASE-T
- 100BASE-TX
- 1000BASE-T

Le module 10 gigabits prend en charge les ports SFP+ (form-factor pluggable) avec des modules d'interface et des câbles de raccord. La négociation automatique est toujours activée :

10GBASE-SR
10GBASE-LR

Module Ethernet de gauche

A huit ports Ethernet à paire torsadée non protégée (RJ45). Les numéros Ethernet vont de ETH0 à ETH7 et correspondent au nombre de ports disponibles.

Module Ethernet de droite

A deux ports SFP à 10 gigabits. Les numéros Ethernet vont de ETH8 à ETH9 et correspondent au nombre de ports disponibles.

Voir Configuration du réseau Ethernet pour une description de la numérotation Ethernet.

Remarque : Les modules Ethernet ne sont pas remplaçables à chaud. Le remplacement à chaud des modules provoque une panne du système et peut éventuellement endommager votre dispositif.

Modules d'unité de disque dur

Le panneau avant du dispositif inclut quatre modules d'unité de disque dur de 2,5 pouces. Le dispositif prend en charge les unités de disque dur SAS et il y a deux voyants sur chaque module d'unité de disque dur. Le voyant gauche témoigne de l'activité du disque dur et le voyant droit indique un problème potentiel :

- Un voyant vert clignotant indique que des accès à l'unité de disque dur se produisent.
- Un voyant ambre clignotant indique que l'unité de disque dur est en erreur.
- Aucun voyant allumé indique que l'unité de disque dur n'est pas active.

Remarque : Les modules d'unité de disque dur ne sont pas remplaçables à chaud. Le remplacement à chaud des modules peut provoquer une panne du système.

Rubrique parent : Spécifications et fonctions du dispositif type 7199-92x

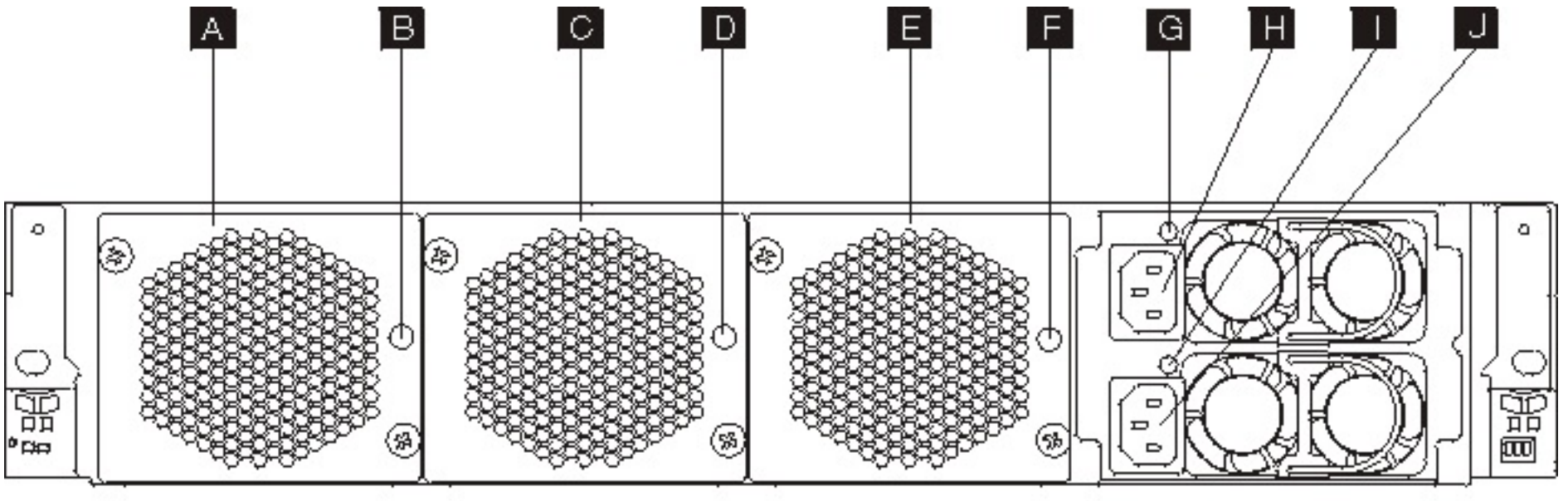
[1] Un périphérique simple qui transmet (entrées) et reçoit (sorties) des données ASCII.

Vue arrière du type 7199-92x

La vue arrière montre les composants et les voyants situés à l'arrière du dispositif. Les modules de ventilation et les modules d'alimentation sont installés depuis l'arrière du dispositif.

Diagramme de la vue arrière

Figure 1. Vue arrière du type 7199-92x



Les annotations du diagramme correspondent aux composants suivants du panneau arrière d'un dispositif de type 7199-92x :

A

Module de ventilation 1

B

Voyant du module de ventilation 1

C

Module de ventilation 2

D

Voyant du module de ventilation 2

E

Module de ventilation 3

F

Voyant du module de ventilation 3

G

Voyant du module d'alimentation 1

H

Module d'alimentation 1

E

Voyant du module d'alimentation 2

Identité J2EE (J

Module d'alimentation 2

Modules de ventilation

Le dispositif comprend trois modules de ventilation. Chaque module de ventilation contient un ventilateur de refroidissement individuel avec un voyant dans chaque module de ventilation :

- Si le voyant ambre est allumé, cela signifie qu'il y a un problème avec le module de ventilation.
- Si le voyant ambre n'est pas allumé, cela signifie que les ventilateurs fonctionnent normalement.

La vitesse des ventilateurs dépend de la température du dispositif. Quand la température augmente, la vitesse des ventilateurs augmente de façon à maintenir une température équilibrée pour le dispositif.

Modules d'alimentation

Le dispositif est alimenté par deux modules d'alimentation redondants. Un seul module d'alimentation suffit pour une prise en charge des opérations du dispositif. Les modules d'alimentation sont remplaçables à chaud : vous pouvez donc remplacer l'un de ceux-ci sans devoir mettre hors tension le dispositif. Chaque module d'alimentation comporte un voyant :

- Si le voyant d'alimentation ambre est allumé, c'est que l'alimentation a généré une erreur.
- Si le voyant ambre n'est pas allumé, cela signifie que le module d'alimentation fonctionne normalement.

Remarque : Lorsque vous appuyez sur le bouton d'alimentation pour éteindre le dispositif, le courant électrique continue d'y arriver. Pour couper complètement l'électricité au dispositif, déconnectez tous les câbles d'alimentation.

Rubrique parent : Spécifications et fonctions du dispositif type 7199-92x

Configuration du réseau Ethernet

Les modules Ethernet étendent les options de connectivité réseau. Chaque dispositif a deux modules Ethernet. Les modules Ethernet sont numérotés de gauche à droite, mais si un module a moins de huit ports, il utilisera le numéro de port le plus petit de la plage.

Convention de numérotation

La convention de numérotation pour la configuration des interfaces Ethernet et l'installation des câbles réseau est la suivante :

- Le module de gauche va de `ETH0` à `ETH7`
- Le module de droite va de `ETH8` à `ETH9`

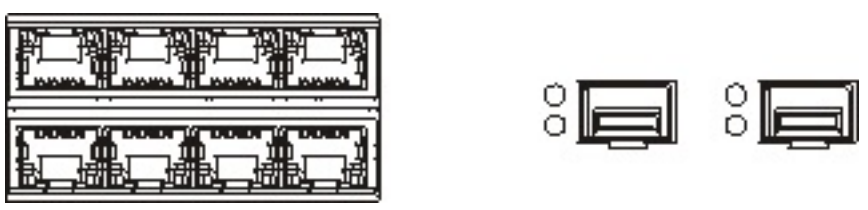
Connexions du type 7199

Chaque module Ethernet a une des configurations suivantes :

- Le module Ethernet de gauche a huit ports Ethernet à 1 gigabits, qui sont des connecteurs RJ45.
- Le module Ethernet de droite a deux ports Ethernet à 10 gigabits, qui sont des émetteurs-récepteurs SFP+ (small form-factor pluggable).

Le dispositif a dix connexions Ethernet. Les noms des interfaces Ethernet sont `ETH0` à `ETH7`, `ETH8` et `ETH9`.

Figure 1. Connexion Ethernet 8x2



Rubrique parent : Spécifications et fonctions du dispositif type 7199-92x

Interopérabilité du produit

WebSphere DataPower XC10 Appliance a été testé du point de vue de l'interopérabilité avec d'autres produits IBM®.

WebSphere Application Server

WebSphere Application Server doit implémenter des scénarios de session HTTP et de grille de données de mémoire cache dynamique. Pour plus d'informations sur les éditions spécifiques de WebSphere Application Server qui sont requises, voir Configuration requise.

WebSphere DataPower Integration Appliance XI50

Vous pouvez utiliser WebSphere DataPower XC10 Appliance en tant que cache secondaire avec WebSphere DataPower Integration Appliance XI50. La passerelle REST rend possible cette intégration en permettant aux clients non-Java d'accéder à des grilles de données simples avec un ensemble d'opérations HTTP. Pour plus d'informations sur la passerelle REST, voir Développement d'applications de grille de données avec la passerelle REST. Pour des informations et un exemple d'application montrant l'intégration entre WebSphere DataPower XC10 Appliance et WebSphere DataPower Integration Appliance XI50, voir Utilisation de WebSphere DataPower XC10 Appliance en tant que cache secondaire pour WebSphere DataPower XI50 Integration Appliance.

WebSphere Portal

Vous pouvez rendre persistantes des sessions HTTP à partir de WebSphere Portal dans une grille de données sur le dispositif. Pour plus d'informations sur la création de cette configuration, voir Configuration du gestionnaire de sessions HTTP avec WebSphere Portal. En outre, IBM Web Content Manager dans IBM WebSphere Portal peut utiliser des instances de mémoire cache dynamique pour stocker du contenu qui est extrait du gestionnaire de contenu Web lorsque la mise en cache avancée est activée. Le dispositif de mise en cache de WebSphere DataPower XC10 Appliance propose une implémentation de cache dynamique qui stocke le contenu mis en cache dans une grille de données élastique au lieu d'utiliser l'implémentation de mise en cache dynamique par défaut.

WebSphere Commerce

WebSphere Commerce Version 7.0.0.1 prend désormais en charge l'utilisation de WebSphere eXtreme Scale Client Version 7.1. Vous pouvez utiliser WebSphere DataPower XC10 Appliance pour mettre en mémoire cache les données du cache dynamique de WebSphere Commerce. Pour plus d'informations, voir Création de grilles de données en cache dynamique.

Tivoli Monitoring

Lorsque vous activez la surveillance SNMP pour WebSphere DataPower XC10 Appliance, vous pouvez importer ces données dans IBM Tivoli Monitoring version 6.2.2 groupe de correctifs 2 ou version ultérieure. Tivoli Monitoring permet de surveiller et gérer des applications système et réseau sur plusieurs systèmes d'exploitation, de contrôler la disponibilité et les performances de toutes les parties de votre entreprise et de générer des rapports pour le suivi des tendances et la résolution des incidents. Pour plus d'informations, voir Importing SNMP data from WebSphere DataPower XC10 Appliance into Tivoli Monitoring .

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Information associée:

Utilisation de WebSphere eXtreme Scale pour améliorer les performance WebSphere Portal et d'IBM Web Content Manager

.NET

Remarques relatives à Microsoft .NET

Deux environnements .NET existent dans WebSphere eXtreme Scale : l'environnement de développement et l'environnement d'exécution. Ces environnements ont des exigences spécifiques.

Configuration requise pour l'environnement de développement

Version de Microsoft .NET

La version .NET 3.5 et les versions suivantes sont prises en charge.

Microsoft Visual studio

Vous pouvez utiliser les versions suivantes de Visual Studio :

- Visual Studio 2008 SP1
- Visual Studio 2010 SP1
- Visual Studio 2012

Windows

Toute version de Windows compatible avec l'édition de Visual Studio que vous utilisez est compatible. Voir les liens suivants pour plus d'informations sur la configuration Windows requise pour Visual Studio :

- Configuration système requise pour Visual Studio 2008
- Configuration système requise pour Visual Studio 2010 Professional
- Configuration système requise pour Visual Studio 2012 Professional

Mémoire

- 1 Go (pour les installations 32 bits et 64 bits)

Espace disque

WebSphere eXtreme Scale nécessite 50 Mo d'espace disque disponible en plus des exigences Visual Studio.

Environnement d'exécution

Version de Microsoft .NET

La version .NET 3.5 et les versions suivantes sont compatibles, y compris l'exécution dans un environnement .NET 4.0 uniquement.

Windows

Tout environnement Windows qui remplit les exigences de version Microsoft .NET répertoriées plus haut.

Mémoire

65 Mo par processus qui accède aux données stockées sur les serveurs WebSphere eXtreme Scale.

Espace disque

WebSphere eXtreme Scale nécessite 35 Mo d'espace disque disponible. Lorsque la fonction de trace est activée, un espace disque supplémentaire de 2,5 Go est nécessaire.

Exécution d'WebSphere eXtreme Scale

Vous devez utiliser le mécanisme de transport eXtremeIO lorsque vous utilisez des applications client .NET. Pour plus d'informations sur eXtremeIO, voir Configuration d'IBM eXtremeIO (XIO).

2.5+ Configuration requise pour le fournisseur de stockage d'état de session ASP.NET

- Le fournisseur de stockage de session ASP.NET requiert l'une des versions de serveur IIS suivantes :
  - IIS 6.0 (fourni avec Windows Server 2003)
  - IIS 7.0 (fourni avec Windows Server 2008)
  - IIS 7.5 (fourni avec Windows Server 2008 R2)
  - IIS 8.0 (fourni avec Windows Server 2012)
- Mémoire : 120 Mo supplémentaires par processus (mémoire totale de 185 Mo).
- Sécurité : l'ID associé au pool d'applications ASP.NET doit disposer des droits d'administrateur.
- Sécurité : le niveau de confiance doit être Complet pour l'application ASP.NET.

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Installation de WebSphere DataPower XC10 Appliance

Pour installer WebSphere DataPower XC10 Appliance au sein de votre environnement existant, vous devez dans un premier temps installer le matériel du dispositif. Installez ensuite WebSphere eXtreme Scale Client dans votre environnement d'application.

Démarrage rapide : Installation du matériel du dispositif
Avant de pouvoir commencer à utiliser le dispositif, vous devez l'installer dans l'armoire, configurer les accès nécessaires, démarrer l'interface utilisateur et vérifier l'état opérationnel du dispositif.

Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance
Une fois le dispositif DataPower XC10 Appliance physiquement installé et connecté, il est prêt à être initialisé et configuré.

Installation de WebSphere eXtreme Scale Client
Pour que IBM® WebSphere DataPower XC10 Appliance fonctionne avec WebSphere Application Server ou des clients dans un environnement autonome, vous devez installer WebSphere eXtreme Scale Client ou intégrer les fichiers JAR (Java™ ARchive) client à votre application.

Installation de profil Liberty
Vous installez l'environnement de service d'applications profil Liberty en utilisant le gestionnaire d'installation ou en exécutant un fichier d'archive Java (fichier JAR).

Identification et résolution des incidents liés à l'installation du produit
IBM Installation Manager est un programme d'installation commun à de nombreux logiciels IBM qui vous permet d'installer cette version de WebSphere eXtreme Scale.

Démarrage rapide : Installation du matériel du dispositif

Avant de pouvoir commencer à utiliser le dispositif, vous devez l'installer dans l'armoire, configurer les accès nécessaires, démarrer l'interface utilisateur et vérifier l'état opérationnel du dispositif.

Avant de commencer

- Vous devez disposer d'une connexion réseau à 1 gigabit pour le port de gestion `MGMT0`.
- Déterminez si vous utilisez les ports 1 gigabit ou 10 gigabits pour votre grille de données. Vous devez utiliser le module Ethernet de gauche (ports 1 gigabit) ou le module Ethernet de droite (ports 10 gigabits).
- Vous devez connaître les adresses IP des interfaces Ethernet pour l'accès à la gestion et l'utilisation de la grille de données.
- Vous devez connaître l'adresse IP des passerelles par défaut (routeurs) prenant en charge les sous-réseaux des interfaces Ethernet.
- Vous devez connaître l'adresse IP pour les services réseau (SSH, Telnet, etc.).

Procédure

1. Installez physiquement le dispositif dans l'armoire.

   Important : N'essayez pas d'ouvrir le boîtier du dispositif. L'ouverture du boîtier déclenche une anomalie de sécurité et le dispositif cesse alors de fonctionner. Dans ce cas, le dispositif devra être retourné à IBM® pour réinitialisation.

   a. Déballez le dispositif avec précaution. Localisez tous les cordons d'alimentation, câbles série et rails fournis.
   b. Identifiez l'emplacement d'installation du dispositif et assurez-vous que l'espace disponible au-dessus et au-dessous de celui-ci est suffisant pour sa ventilation et sa maintenance.
   c. Solidarisez les glissières de montage.
   d. Installez le dispositif sur les rails et faites-le glisser à sa place.
   e. En façade du dispositif, utilisez les câbles réseau pour connecter le dispositif à votre réseau. Les câbles Ethernet ne sont pas fournis.
      - Connectez le port de gestion `MGMT0` à un réseau à 1 gigabit.
      - Pour le trafic autre que le trafic de gestion pour la grille de données, utilisez le module Ethernet de gauche (ports 1 gigabit) ou le module Ethernet de droite (ports 10 gigabits).
   f. Utilisez les cordons d'alimentation fournis pour connecter les alimentations électriques aux prises de courant.
   g. A partir d'une console série, établissez une connexion avec le connecteur `CONSOLE` situé à l'avant du dispositif, en la configurant sur 9600 bauds 8N1 (8 bits par caractère, pas de parité, 1 bit d'arrêt) et sans contrôle de flux. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion. Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble. L'émulateur de terminal recommandé pour la console série est VT100.
   h. Appuyez sur le bouton d'alimentation. Le voyant d'alimentation vert s'allume et la console en série s'affiche.
   i. L'invite de connexion s'affiche. Pour la configuration initiale du dispositif, connectez-vous avec l'ID utilisateur et le mot de passe : xcadmin/xcadmin.

      Important : Ne perdez pas l'ID utilisateur et le mot de passe xcadmin. Si vous perdez ces informations, vous ne pouvez pas vous reconnecter au dispositif et vous devez le renvoyer à IBM pour réinitialisation ; toutes ses données seront alors effacées. Pour garantir l'accessibilité de l'ID utilisateur et du mot de passe xcadmin, vous pouvez configurer un serveur SMTP et une adresse électronique afin de permettre la réinitialisation du mot de passe xcadmin.

2. Configurez les accès au dispositif à l'aide de la console en série. Un assistant vous guide à travers le processus d'acceptation des contrats de licence et de configuration des ports Ethernet. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.
3. Mettez à jour le microprogramme de votre dispositif vers la version la plus récente. Pour plus d'informations, voir Mise à jour du microprogramme.
4. Suivez l'état de démarrage du dispositif après la mise à niveau du microprogramme. Vous pouvez suivre l'état de démarrage du dispositif dans l'interface utilisateur. Dans la barre d'adresse du navigateur Web, entrez : `https://<nom_hôte_dispositif>:9443/`. Vous pouvez également suivre la progression du démarrage en exécutant la commande **`start-progress`** dans le terminal que vous utilisez pour initialiser le dispositif. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.

5. Démarrez l'interface utilisateur.
    a. Dans la barre d'adresse du navigateur Web, entrez l'URL et le port définis lors de l'initialisation de l'unité. Vous pouvez utiliser l'adresse IP que vous avez définie ou le nom d'hôte qui correspond à l'adresse IP, par exemple `https://myXC10.ibm.com`. Utilisez le protocole HTTP sécurisé HTTPS.
    b. Entrez `xcadmin` dans la zone **Utilisateur** .
    c. Entrez le mot de passe correspondant dans la zone **Mot de passe**. Par défaut, ce mot de passe est `xcadmin`.
    d. Cliquez sur **Connexion**. Pour vous déconnecter, cliquez sur **Déconnexion**.
6. Vérifiez que le dispositif est opérationnel.
    - Le voyant vert sur le panneau frontal du dispositif est allumé.
    - Le voyant ambre d'erreur sur le panneau frontal du dispositif est éteint.
    - Le voyant vert de mise en cache sur le panneau frontal du dispositif doit s'allumer.
    - Le voyant ambre d'erreur de mise en cache sur le panneau frontal du dispositif est éteint.
    - L'afficheur LCD du panneau frontal du dispositif affiche le numéro de version du produit.

    Pour un diagramme montrant l'emplacement de ces voyants sur le dispositif, voir Spécifications et fonctions du dispositif .

    Si vous avez des problèmes, contactez le support IBM. Accédez au site Web : http://www-947.ibm.com/support/entry/portal/overview/software/websphere/websphere_datapower_xc10_appliance .

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Tâches associées:
Mise à jour du microprogramme
Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Une fois le dispositif DataPower XC10 Appliance physiquement installé et connecté, il est prêt à être initialisé et configuré.

Avant de commencer

Pour effectuer la configuration initiale, vous devez utiliser une connexion série. Cette connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif. Si vous utilisez un PC comme console série, vous devez utiliser un programme de communication série conçu pour un PC fonctionnant sous Windows ou Linux. Vous pouvez utiliser un matériel dédié, par exemple une console VT100, ou un PC exécutant un émulateur comme HyperTerminal ou Minicom. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion avec le dispositif.

Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de pouvoir continuer.

Avant de définir la configuration de base, rassemblez les informations suivantes :

- Assurez-vous que le port de gestion `MGMT0` dispose d'une connexion réseau d'un débit de 1 gigabit.
- Déterminez si vous utilisez les ports 1 gigabit ou 10 gigabits pour votre grille de données. Vous devez utiliser le module Ethernet de gauche (ports 1 gigabit) ou le module Ethernet de droite (ports 10 gigabits).
- Adresse IP et masque de sous-réseau de l'interface Ethernet pour l'accès à la gestion du dispositif (`MGMT`).
- Adresse IP des passerelles par défaut (routeurs) prenant en charge les sous-réseaux des interfaces Ethernet.
- Adresse IP du serveur DNS (Domain Name System).
- Informations sur le serveur de messagerie (pour la configuration des notifications par courrier électronique). Pour plus d'informations, voir Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance.
- Informations sur le serveur NTP (Network Time Protocol) pour permettre les communications entre les dispositifs de la collectivité.

Pourquoi et quand exécuter cette tâche

Utilisez cette tâche afin d'initialiser votre DataPower XC10 Appliance pour la première fois. La procédure à suivre pour effectuer votre première connexion au dispositif est légèrement différente de celle que vous utiliserez lors des connexions ultérieures.

ATTENTION :

1. **Consultez les informations importantes sur la protection de l'ID utilisateur et du mot de passe xcadmin qui se trouvent dans la rubrique Mot de passe xcadmin.**
2. **N'essayez pas d'ouvrir le boîtier du dispositif. L'ouverture du boîtier déclenche une anomalie de sécurité et le dispositif cesse alors de fonctionner. Dans ce cas, vous devez exécuter la commande `device clear-intrusion` pour rétablir le fonctionnement du dispositif.**
3. **Ne perdez pas le câble série fourni avec le dispositif. Vous en aurez besoin pour la configuration initiale du dispositif et peut-être ultérieurement pour l'identification des incidents. Ce câble a été spécialement conçu pour fonctionner avec le dispositif. Un autre câble ne fonctionnerait pas forcément avec le dispositif.**

Effectuez la configuration initiale de base du microprogramme. Il s'agit ici de la configuration minimale en vue d'ajouter un système WebSphere DataPower XC10 Appliance à votre environnement.

Procédure

1. Initialisez le dispositif. Procédez comme suit :
    a. Connectez la console série au dispositif à l'aide du câble série ou du câble USB/série fourni. Vous devez connecter le câble au connecteur CONSOLE situé sur la face avant du dispositif et le terminal ASCII ou le PC exécutant un logiciel d'émulation de terminal doit être opérationnel. Cette connexion vous permet voir les messages émis par le dispositif lors de son démarrage. Configurez le logiciel d'émulation avec 9600 baud 8N1 (8 bits par caractère, pas de parité, 1 bit d'arrêt) et sans contrôle de flux. L'émulateur de terminal recommandé pour la console série est VT100.
    b. Assurez-vous que le dispositif est sous tension. Si le dispositif est hors tension, appuyez sur le bouton d'alimentation. Le bouton d'alimentation se trouve sur le panneau frontal du dispositif.

Patientez quelques secondes pendant le démarrage du dispositif. Une fois le dispositif sous tension :

- Le voyant vert s'allume sur le panneau frontal du dispositif et le ventilateur se met en marche.
- L'invite de connexion s'affiche dans la console série. Pour la configuration initiale du dispositif, connectez-vous avec l'ID utilisateur et le mot de passe par défaut : xcadmin/xcadmin.

c. Acceptez les contrats de licence WebSphere DataPower XC10 Appliance. La première fois, avant de poursuivre, vous devez accepter les licences. Saisissez `Accept`, `Reject²` ou `StartOver` à chaque invite de licence.

d. Configurez le port Ethernet `MGMT`. Spécifiez l'adresse IP au format de routage interdomaine sans classes (CIDR).

e. Configurez la passerelle par défaut du port Ethernet `MGMT`.

f. Configurez les ports Ethernet pour votre grille de données. Spécifiez si vous utilisez des ports 1 gigabit ou des ports 10 gigabits. Configurez l'adresse CIDR pour les ports applicables.

g. Configurez les serveurs DNS (Domain Name System). Spécifiez une adresse IP valide pour votre serveur DNS.

2. Mettez à jour le microprogramme du dispositif. Pour télécharger ou mettre à jour le microprogramme de WebSphere DataPower XC10 Appliance, vous devez disposer des droits d'administration du dispositif. Le dispositif ne doit pas nécessairement être lui-même connecté à Internet pour extraire la mise à jour du microprogramme. Pour plus d'informations sur le téléchargement ou la mise à jour du microprogramme, reportez-vous à la rubrique Mise à jour du microprogramme.

3. Si vous installez le microprogramme pour la première fois sur un nouveau dispositif, vous devez exécuter la commande **`clear-all`** sur le dispositif. Effectuez les étapes suivantes après le redémarrage qui suit la mise à niveau du microprogramme :

   a. Etablissez une connexion avec le dispositif en tant qu'utilisateur `xcadmin`. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.

   b. Exécutez la commande **`clear-all`**.

   ```
   Console> clear-all
   Force Stopped all XC-10 processes
   Deleting configuration data and logs
   Deleting grid data
   ```

   ATTENTION :

   N'exécutez aucune autre commande avant la commande `clear-all`. L'exécution d'autres commandes peut créer des problèmes dans la configuration de votre dispositif.

   ATTENTION :

   N'exécutez aucune autre commande pendant que la commande `clear-all` est en cours d'exécution. Si vous voulez surveiller la progression du démarrage, n'utilisez que la commande `start-progress` ou la page correspondante de l'interface utilisateur du dispositif.

   c. Une fois que vous avez exécuté la commande **`clear-all`**, vous pouvez suivre la progression du démarrage du dispositif. Utilisez pour cela l'une des options suivantes :

   - 2.5+ Utilisez l'interface utilisateur du dispositif. Dans la barre d'adresse du navigateur Web, entrez l'URL et le port suivants : `https://<nom-hôte_dispositif>:9443/`. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.
   - Exécutez la commande **`start-progress`**. Lorsque cette commande renvoie `STARTED`, cela signifie que le dispositif est prêt à l'emploi.

4. Pour sécuriser la configuration, modifiez le mot de passe de l'utilisateur xcadmin. Le mot de passe par défaut est `xcadmin`. Vous pouvez modifier ce mot de passe à l'aide de la commande suivante :

   ```
   user password <ancien_mot_de_passe> <nouveau_mot_de_passe>
   ```

5. Vérifiez la configuration. Utilisez l'interface utilisateur avec un navigateur Web pour vérifier la configuration.

   Avertissement : La procédure de vérification ci-dessous suppose que le nom d'hôte de l'interface Ethernet est `myXC10.ibm.com`.

Pour accéder à l'interface utilisateur à partir d'un navigateur, procédez comme suit :

a. Ouvrez un navigateur Web. Ouvrez votre navigateur Web à partir d'un ordinateur connecté au réseau.
b. Entrez l'URL. Dans la barre d'adresse, entrez l'URL défini lors de l'initialisation du dispositif. Par exemple : `https://myXC10.ibm.com`.

   Remarque : Utilisez le protocole `https` et non pas `http`.

c. Connectez-vous au dispositif. Connectez-vous au dispositif à l'aide du compte `XCADMIN` local et du mot de passe. Le mot de passe que vous tapez est en clair ; pour des raisons de sécurité il ne s'affiche donc pas.
d. Cliquez sur **Connexion**.

   Si la page Bienvenue s'affiche, l'authentification du compte `XCADMIN` local a abouti.

6. Effectuez la configuration.

Résultats

La configuration initiale pour WebSphere DataPower XC10 Appliance est terminée.

Que faire ensuite

Configurez les paramètres de votre dispositif dans l'interface utilisateur. Ces paramètres incluent les utilisateurs et groupes d'utilisateurs, la sécurité, la distribution par courrier électronique, les serveurs de noms de domaine (DNS) et les paramètres de date et d'heure. Pour plus d'informations, voir Configuration de votre dispositif.

Mot de passe xcadmin
Après avoir défini l'ID et le mot de passe de l'administrateur (xcadmin), conservez-les dans un endroit sûr. Vous pouvez configurer la distribution par courrier électronique de sorte à permettre la réinitialisation des mots de passe dans l'interface utilisateur.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Tâches associées:
Mise à jour du microprogramme
Démarrage rapide : Installation du matériel du dispositif

Mot de passe xcadmin

Après avoir défini l'ID et le mot de passe de l'administrateur (xcadmin), conservez-les dans un endroit sûr. Vous pouvez configurer la distribution par courrier électronique de sorte à permettre la réinitialisation des mots de passe dans l'interface utilisateur.

Sauvegarde de l'ID utilisateur et du mot de passe

L'ID utilisateur et le mot de passe xcadmin étant nécessaires pour se connecter au dispositif, placez-les en lieu sûr une fois que vous les avez modifiés. Si vous les perdez et que vous n'avez pas moyen de les récupérer, vous devez renvoyer le dispositif à IBM® en vue de sa réinitialisation

Réinitialisation des mots de passe par courrier électronique

Si vous configurez la distribution par courrier électronique, tous les utilisateurs peuvent restaurer leur mot de passe en cliquant sur le lien **Mot de passe oublié ?** dans l'écran de connexion de l'interface utilisateur. Un courrier électronique contenant un nouveau mot de passe généré est envoyé à l'utilisateur.

ATTENTION :

La seule manière de réinitialiser le mot de passe xcadmin est d'utiliser le lien Mot de passe oublié ? dans l'écran de connexion de l'interface utilisateur. Si vous oubliez le mot de passe et que la distribution par courrier électronique n'est pas configurée, vous devez réinitialiser le dispositif avec la commande `device RESET`, ce qui réinitialise tous les paramètres du dispositif. Voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif pour plus d'informations sur la commande `device RESET`.

Modification du mot de passe xcadmin

Vous pouvez éditer le mot de passe de xcadmin dans l'interface utilisateur ou l'interface de ligne de commande du dispositif.

Pour modifier le mot de passe dans l'interface utilisateur, éditez l'utilisateur. Pour plus d'informations, voir Gestion des utilisateurs.

Pour modifier le mot de passe dans l'interface de ligne de commande du dispositif, vous pouvez utiliser la commande **`user password`**. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Gestion du serveur DNS (Domain Name System)
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance

Référence associée:
Passerelle REST : Configuration de la sécurité

Installation de WebSphere eXtreme Scale Client

Pour que IBM® WebSphere DataPower XC10 Appliance fonctionne avec WebSphere Application Server ou des clients dans un environnement autonome, vous devez installer WebSphere eXtreme Scale Client ou intégrer les fichiers JAR (Java™ ARchive) client à votre application.

Pourquoi et quand exécuter cette tâche

L'installation de WebSphere eXtreme Scale Client est requise pour que vos applications client communiquent avec IBM WebSphere DataPower XC10 Appliance. Avant d'installer le client, vous devez déterminer si vous souhaitez l'installer dans un environnement autonome ou dans un environnement WebSphere Application Server.

Si vos applications utilisent les sessions HTTP ou la mémoire cache dynamique de WebSphere Application Server, vous devez installer WebSphere eXtreme Scale Client dans l'environnement WebSphere Application Server.

Des grilles de données simples peuvent être utilisées dans un environnement WebSphere Application Server imbriqué ou dans un environnement autonome. Lors de l'installation du client dans un environnement autonome, l'installation n'utilise pas WebSphere Application Server.

Pour plus d'informations sur les types de grille de données, voir Topologie du dispositif : collectivités, zones et grilles de données.

Procédure

1. Téléchargez et installez IBM Installation Manager et vérifiez que vous avez installé les référentiels de produit nécessaires. Voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.
2. Téléchargez WebSphere eXtreme Scale Client à partir du site de support. Pour plus d'informations sur l'emplacement de téléchargement du client, voir le Portail de support.
3. Exécutez l'installation à l'aide d'Installation Manager. Choisissez l'offre de produit correcte. Les offres de produit sont disponibles selon les référentiels que vous avez ajoutés à vos préférences d'installation dans Installation Manager. Les offres de produit disponibles pour WebSphere eXtreme Scale Client sont les suivantes :
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 6
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

   Utilisez l'installation qui convient à votre configuration planifiée :
   - **Installation client imbriquée** : Si vous utilisez la mémoire cache dynamique ou les sessions HTTP, vous devez utiliser l'installation imbriquée. Vous pouvez également utiliser l'installation imbriquée avec des grilles de données simples. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
   - **Installation client autonome** : Vous ne pouvez utiliser l'installation client que dans un environnement autonome comportant des grilles de données simples. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.
   - **Installation en mode silencieux** : Vous pouvez également installer le client dans un environnement autonome ou dans un environnement WebSphere Application Server imbriqué, à l'aide d'un fichier de réponses. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses.

Que faire ensuite

Configurez le dispositif. Pour plus d'informations, voir Configuration de votre dispositif.

Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client
Les offres de produit WebSphere eXtreme Scale Client sont disponibles dans les référentiels du produit. Avant de pouvoir accéder à ces référentiels, vous devez installer IBM Installation Manager.

Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager
Utilisez IBM Installation Manager pour désinstaller des offres de produit WebSphere eXtreme Scale Client.

.NET 2.5+ **Présentation de l'installation de WebSphere eXtreme Scale Client for .NET**
Vous pouvez installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution ou dans un environnement d'exécution et de production.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Java

Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Les offres de produit WebSphere eXtreme Scale Client sont disponibles dans les référentiels du produit. Avant de pouvoir accéder à ces référentiels, vous devez installer IBM® Installation Manager.

Vous pouvez installer Installation Manager en utilisant les fichiers disponibles sur le support du produit, ou en utilisant un fichier obtenu à partir du site Passport Advantage ou en utilisant un fichier disponible sur le site Web de téléchargement d'IBM Installation Manager. Un fichier est un fichier compressé contenant des images d'installation.

Remarque :

Installation Manager est disponible pour le téléchargement en version 32 et 64 bits. Vous pouvez utiliser une version d'Installation Manager pour installer WebSphere eXtreme Scale.

Installation Manager vous permet d'accéder aux référentiels de produit nécessaires. Vous devez accéder à ces référentiels afin d'installer les offres de produit WebSphere eXtreme Scale.

Vous disposez de deux options pour accéder aux référentiels de produit.

Option 1 : Accédez aux référentiels de produit sur le support physique et utilisez une installation locale

1. Installez Installation Manager sur votre système.
2. Utilisez Installation Manager pour installer l'offre de produit à partir des référentiels du produit présents sur le support.

Option 2 : Téléchargez les référentiels de produit à partir de Passport Advantage et utilisez une installation locale

1. Téléchargez les référentiels à partir du site Passport Advantage.

   Remarque : Voir Voir Logiciels pris en charge pour obtenir la liste des images d'installation IBM WebSphere eXtreme Scale téléchargeables depuis le site Web IBM Passport Advantage et d'autres informations.
2. Installez Installation Manager sur votre système.
3. Utilisez Installation Manager pour installer le produit à partir des référentiels de produit téléchargés.

ID d'offre pour les produits WebSphere eXtreme Scale Client
Lors de l'installation des mises à jour de produit ou lors de l'annulation de correctifs, vous êtes tenu d'indiquer l'ID d'offre à partir de la ligne de commande. Utilisez le tableau ci-dessous pour identifier l'offre de produit.

Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server
Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement dans lequel WebSphere Application Server ou WebSphere Application Server Network Deployment est installé. Vous pouvez utiliser les fonctions existantes de WebSphere Application Server ou WebSphere Application Server Network Deployment pour améliorer vos applications WebSphere DataPower XC10 Appliance.

Installation d'IBM Installation Manager à l'aide de l'interface graphique
Pour accéder aux référentiels de produit nécessaires à l'installation des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM Installation Manager. Vous pouvez installer Installation Manager à l'aide d'une console d'assistant.

Installation d'IBM Installation Manager à l'aide de la ligne de commande
Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM Installation Manager. Vous pouvez installer Installation Manager à partir de la ligne de commande.

Installation d'IBM Installation Manager à l'aide de fichiers de réponses
Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM Installation Manager. Vous pouvez installer Installation Manager à l'aide de fichiers de réponses.

Rubrique parent : Installation de WebSphere eXtreme Scale Client

ID d'offre pour les produits WebSphere eXtreme Scale Client

Lors de l'installation des mises à jour de produit ou lors de l'annulation de correctifs, vous êtes tenu d'indiquer l'ID d'offre à partir de la ligne de commande. Utilisez le tableau ci-dessous pour identifier l'offre de produit.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server

Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement dans lequel WebSphere Application Server ou WebSphere Application Server Network Deployment est installé. Vous pouvez utiliser les fonctions existantes de WebSphere Application Server ou WebSphere Application Server Network Deployment pour améliorer vos applications WebSphere DataPower XC10 Appliance.

Avant de commencer

- Vérifiez que le répertoire d'installation cible ne contient pas une installation de WebSphere eXtreme Scale Client.
- Arrêtez tous les processus en cours d'exécution dans votre environnement WebSphere Application Server ou WebSphere Application Server Network Deployment. Voir Utilitaires de ligne de commande pour plus d'informations sur les commandes **stopManager**, **stopNode** et **stopServer**.

  ATTENTION :

  Vérifiez que les processus actifs sont arrêtés. Si les processus en cours d'exécution ne sont pas arrêtés, l'installation se poursuit avec des résultats imprévisibles. Sur certaines plateformes, l'installation peut prendre fin dans un état indéterminé.

Important : Lorsque vous installez WebSphere eXtreme Scale Client, ce doit être dans le répertoire dans lequel vous avez installé WebSphere Application Server. Par exemple, si vous avez installé WebSphere Application Server dans `C:\racine_was`, vous devez également choisir `C:racine_was` comme répertoire cible lors de l'installation de WebSphere eXtreme Scale Client.

Pourquoi et quand exécuter cette tâche

Intégrez eXtreme Scale à WebSphere Application Server ou WebSphere Application Server Network Deployment pour appliquer les fonctions d'eXtreme Scale à vos applications Java™ Platform, Enterprise Edition. Les applications Java EE y accèdent à l'aide d'une connexion client.

Procédure

- Pour installer WebSphere eXtreme Scale Client dans un environnement WebSphere Application Server version 8, procédez comme suit :
  1. Installez IBM Installation Manager. Pour plus d'informations, voir Installation d'IBM Installation Manager à l'aide de l'interface graphique.
  2. A l'aide d'Installation Manager, installez l'offre de produit eXtreme Scale appropriée :
     - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

     Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.
  3. Téléchargez les référentiels WebSphere Application Server version 8 nécessaires à partir du site Passport Advantage. Pour plus d'informations, voir How to download WebSphere Application Server - Express V8.5 from Passport Advantage.
  4. Installez WebSphere Application Server version 8. Pour plus d'informations, voir Installation du produit sur des systèmes d'exploitation distribués à l'aide de l'interface graphique.
- Pour installer WebSphere eXtreme Scale Client dans un environnement WebSphere Application Server version 7, procédez comme suit :
  1. Installez IBM Installation Manager. Pour plus d'informations, voir Installation d'IBM Installation Manager à l'aide de l'interface graphique.
  2. Installez WebSphere Application Server Version 7 en utilisant le programme d'installation InstallShield MultiPlatform (ISMP). Pour plus d'informations, voir Installation de l'environnement de service d'applications.
  3. Après l'installation, vous devez importer WebSphere Application Server version 7 dans le gestionnaire d'installation pour terminer l'installation. En important WebSphere Application Server version 7 dans Installation Manager, vous pouvez gérer et installer des groupes de correctifs pour le produit à partir d'un seul emplacement. Vous devez veiller à ce que les référentiels nécessaires soient configurés dans Installation Manager pour l'accès aux groupes de correctifs et aux mises à jour. Pour plus d'informations sur l'importation d'une installation existante de WebSphere Application Server 7 dans Installation Manager, voir Importation des infos produit d'IBM WebSphere Application Server dans le registre d'IBM Installation Manager.
  4. A l'aide d'Installation Manager, installez l'offre de produit eXtreme Scale appropriée :

- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7

Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation d'IBM Installation Manager à l'aide de l'interface graphique

Pour accéder aux référentiels de produit nécessaires à l'installation des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM® Installation Manager. Vous pouvez installer Installation Manager à l'aide d'une console d'assistant.

Avant de commencer

Vous devez installer IBM Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Accédez à l'emplacement contenant les fichiers d'installation du gestionnaire d'installation et exécutez l'une des commandes suivantes :

   Installation par un administrateur :

   - Windows `install.exe`
   - UNIX Linux `./install`

   Installation par un non administrateur :

   - Windows `userinst.exe`
   - UNIX Linux `./userinst`

   Pour plus d'informations sur les installations administratives et non administratives, voir Installation en tant qu'administrateur, non-administrateur ou groupe

   Installation en mode groupe :

   UNIX Linux `./groupinst`

   Remarques sur le mode groupe :

   - Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des progiciels.

     Le mode Groupe ne signifie pas que deux personnes peuvent utiliser la même instance d'IBM Installation Manager en même temps.
   - Windows Le mode groupe n'est pas disponible sur les systèmes d'exploitation Windows.
   - Si vous n'installez pas le gestionnaire d'installation avec le mode Groupe, vous ne pourrez pas utiliser le mode Groupe pour gérer les produits que vous installez ultérieurement à l'aide de cette instance d'Installation Manager.
   - Remplacez l'emplacement d'installation par défaut de l'utilisateur en cours par un emplacement accessible à tous les utilisateurs du groupe.
   - Configurez vos groupes, droits d'accès et variables d'environnement, comme décrit dans les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5 avant d'effectuer l'installation en mode groupe.
   - Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

2. Veillez à sélectionner le package du gestionnaire d'installation et cliquez sur **Next**.
3. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
4. Cliquez sur **Suivant**.
5. Lisez le récapitulatif et cliquez sur **Installer**. Si l'installation a réussi, le programme affichera un message pour confirmer le succès de l'installation. Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
6. Ajoutez le référentiel du produit à vos préférences Installation Manager.
   a. Démarrez Installation Manager.
   b. Dans le menu du haut, cliquez sur **Fichier > Préférences**.
   c. Sélectionnez **Repositories**.
   d. Cliquez sur **Add Repository**.
   e. Entrez le chemin du fichier `repository.config` dans l'emplacement contenant les fichiers du référentiel :

- Windows `C:\repositories\nom_produit\local-repositories`
- UNIX Linux `/var/repositories/nom_produit/local-repositories`

f. Cliquez sur **OK**.

7. Effacez tous les emplacements énumérés dans la fenêtre Repositories, qui ne seront pas utilisés.
8. Cliquez sur **Apply**.
9. Cliquez sur **OK**.
10. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Que faire ensuite

Lorsque l'installation du gestionnaire d'installation et la configuration du référentiel aboutissent, vous pouvez continuer d'installer WebSphere eXtreme Scale Client ou for pour l'offre de produit. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique
Utilisez Installation Manager à partir de la console de l'assistant pour installer les offres de produit WebSphere eXtreme Scale Client.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Utilisez Installation Manager à partir de la console de l'assistant pour installer les offres de produit WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Démarrez Installation Manager.

   UNIX Linux **Conseil :** Vous pouvez démarrer Installation Manager en mode groupe avec la commande **./IBMIM**.

   - Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des packages logiciels.
   - Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

2. Cliquez sur **Install**.

   Remarque : Si vous êtes invité à vous authentifier, utilisez l'ID et le mot de passe IBM avec lesquels vous vous êtes enregistré sur le site Web du programme.

   Installation Manager analyse ses référentiels définis à la recherche des packages disponibles.

3. Sélectionnez l'une des offres de produit ci-dessous ainsi que la version appropriée.
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

   Si le produit est déjà installé sur votre système, un message s'affichera pour vous en informer. Pour créer une autre installation du produit dans un autre emplacement, cliquez sur **Continuer**.

   Conseil : Si l'option **Search service repositories during installation and updates** est sélectionnée sur la page des préférences de référentiel dans Installation Manager et que vous êtes connecté à Internet, vous pouvez cliquer sur **Check for Other Versions and Extensions**. Ainsi, vous pouvez rechercher les mises à jour dans les référentiels de mise à jour par défaut pour les packages sélectionnés. Dans ce cas, il n'est pas nécessaire d'ajouter l'URL de référentiel de service spécifique à la page des préférences en matière de référentiel d'Installation Manager.

   a. Sélectionnez les correctifs à installer.

      Les correctifs recommandés sont sélectionnés par défaut.

      Vous pouvez choisir de n'afficher que les correctifs recommandés et de masquer les autres.

   b. Cliquez sur **Suivant**.

      Remarque : Installation Manager peut vous inviter à effectuer une mise à jour vers le dernier niveau d'Installation Manager lors de la connexion au référentiel. Si vous y êtes invité, effectuez la mise à jour vers la nouvelle version avant de poursuivre. Pour plus d'informations sur les mises à jour automatiques, voir le centre de documentation d'IBM Installation Manager Version 1.5.

4. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
5. Indiquez le répertoire principal d'installation du produit.

   Le panneau affiche également le répertoire de ressources partagées et les informations d'espace disque.

   Remarque : La première fois que vous installez un package à l'aide d'Installation Manager, spécifiez le répertoire des ressources partagées. Le répertoire des ressources partagées est l'endroit où les artefacts d'installation se trouvent ; ils peuvent être utilisés par un plusieurs groupe de packages. Utilisez votre unité ayant la taille la plus importante pour cette installation. Vous ne pouvez pas

changer l'emplacement du répertoire tant que vous n'avez pas désinstallé tous les packages.

Restrictions :

- Si vous supprimez l'emplacement cible par défaut et que vous ne renseignez pas une zone du répertoire d'installation, vous ne pouvez pas continuer.
- N'utilisez pas de liens symboliques comme répertoire de destination.

  car ils ne sont pas pris en charge.
- N'utilisez pas de point-virgule dans le nom d'un répertoire.

  Si le répertoire de destination inclut un point-virgule, cela signifie que WebSphere eXtreme Scale n'est pas installé comme prévu.

  Windows Sous Windows, le point-virgule est le caractère utilisé pour construire le chemin d'accès aux classes.

  - Windows La longueur maximale du chemin d'accès sur les systèmes d'exploitation Windows Server 2008, Windows Vista et Windows 7 est de 60 caractères.

6. Cliquez sur **Suivant**.
7. Sélectionnez les langues pour lesquelles un contenu traduit doit être installé.

   L'anglais est toujours sélectionné.
8. Cliquez sur **Suivant**.
9. Sélectionnez les fonctions que vous voulez installer.

   En fonction de l'offre de produit sélectionnée, vous pouvez choisir l'une des fonctions suivantes :
   - Console

     Disponible pour toutes les offres de produit WebSphere eXtreme Scale. Vous pouvez choisir d'installer la console de surveillance. Avec la console Web, vous pouvez générer des graphiques des statistiques actuelles et historiques. Cette console fournit un certain nombre de graphiques pour des présentations générales et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.
   - Samples

     Disponible pour toutes les offres de produit WebSphere eXtreme Scale.
10. Cliquez sur **Suivant**.
11. Lisez le récapitulatif et cliquez sur **Installer**.

    - Si l'installation aboutit, le programme affiche un message indiquant que l'installation a abouti.

      Remarque : Le programme peut également afficher d'importantes instructions post-installation.
    - Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
12. Sélectionnez l'outil que vous souhaitez démarrer à la fin de l'installation.
    - Sélectionnez **Outil de gestion des profils pour créer un profil** si vous souhaitez créer un profil de serveur d'applications avec des paramètres appropriés pour un environnement de production.
    - Sélectionnez **Outil de gestion des profils pour créer un profil de serveur d'applications pour un environnement de développement** si vous souhaitez créer un profil de serveur d'applications avec des paramètres appropriés pour un environnement de développement.

      Remarque : Les paramètres de développement sont adaptés à un environnement de développement au sein desquelles des mises à jour sont effectuées et les ressources systèmes sont au minimum. N'utilisez pas les paramètres de développement pour les serveurs de production.
    - Sélectionnez **Aucun** si vous ne souhaitez pas créer de profil lorsque l'installation est terminée.

    Restriction : L'option de démarrage de l'outil de gestion des profils est disponible uniquement si une version de WebSphere Application Server contenant l'outil de gestion des profils est installée.
13. Cliquez sur **Terminer**.

14. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Rubrique parent : Installation d'IBM Installation Manager à l'aide de l'interface graphique

Tâches associées:
Désinstallation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Java

Installation d'IBM Installation Manager à l'aide de la ligne de commande

Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM® Installation Manager. Vous pouvez installer Installation Manager à partir de la ligne de commande.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Accédez à l'emplacement contenant les fichiers d'installation d'Installation Manager et exécutez une des commandes suivantes :

   Installation par un administrateur :

   - Windows `installc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
   - UNIX Linux `./installc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

   Installation par un non administrateur :

   - Windows `userinstc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
   - UNIX Linux `./userinstc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

   Installation en mode groupe :

   UNIX Linux `./groupinstc -acceptLicense -dataLocation` *`emplacement_données_application`* `-log` *`chemin_et_nom_fichier_journal`* `-installationDirectory` *`rép_Installation_Manager`*

   Remarques sur le mode groupe :

   - Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des packages logiciels.
   - Windows Le mode groupe n'est pas disponible sur les systèmes d'exploitation Windows.
   - Si vous n'installez pas Installation Manager avec le mode groupe, vous ne pourrez pas utiliser le mode groupe pour gérer les produits que vous installez ultérieurement à l'aide de cette instance d'Installation Manager.
   - Assurez-vous de bien changer l'emplacement d'installation et de le faire passer de l'emplacement par défaut dans le répertoire de base de l'utilisateur actuel à un emplacement accessible par tous les utilisateurs du groupe.
   - Configurez vos groupes, droits d'accès et variables d'environnement, comme décrit dans les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5 avant d'effectuer l'installation en mode groupe.
   - Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

2. Facultatif : Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

   Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

   Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

Que faire ensuite

Une fois que l'installation d'Installation Manager et la configuration du référentiel aboutissent, vous pouvez continuer d'installer l'offre de produit WebSphere eXtreme Scale WebSphere eXtreme Scale Client ou WebSphere eXtreme Scale for WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.

Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande
Utilisez Installation Manager à partir de la ligne de commande pour installer les offres de produit WebSphere eXtreme Scale Client.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Utilisez Installation Manager à partir de la ligne de commande pour installer les offres de produit WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Ouvrez une session sur votre système.
2. Accédez au sous-répertoire `eclipse/tools` dans le répertoire où vous avez installé Installation Manager.
3. Vérifiez que le référentiel du produit est disponible.

   Windows

   ```
   imcl.exe listAvailablePackages -repositories référentiel_source
   ```

   UNIX Linux

   ```
   ./imcl listAvailablePackages -repositories référentiel_source
   ```

   Vous devez voir un ou plusieurs niveaux de l'offre.
4. Utilisez la commande **imcl** pour installer le produit.

   Windows

   ```
   imcl.exe install com.ibm.websphere.v85_version_offre,ID_fonction_facultative
     -repositories référentiel_source
     -installationDirectory répertoire_installation
     -sharedResourcesDirectory répertoire_partagé
     -accessRights mode_accès
     -preferences clé_préférence=valeur
     -properties clé_propriété=valeur
     -keyring fichier_de_clés -password mot_de_passe
     -acceptLicense
   ```

   UNIX Linux

   ```
   ./imcl install com.ibm.websphere.version_offre,ID_fonction_facultative
     -repositories référentiel_source
     -installationDirectory répertoire_installation
     -sharedResourcesDirectory répertoire_partagé
     -accessRights mode_accès
     -preferences clé_préférence=valeur
     -properties clé_propriété=valeur
     -keyring fichier_de_clés -password mot_de_passe
     -acceptLicense
   ```

Conseils :

- *ID_offre* correspond à l'ID offre répertorié dans la rubrique ID d'offre pour les produits WebSphere eXtreme Scale Client.
- La *version_offre*, qui peut, le cas échéant, être associée à l'ID offre à l'aide d'un trait de soulignement, est une version spécifique de l'offre à installer (`8.6.0.20110503_0200`, par exemple).

  - Si la *version_offre* n'est **pas** spécifiée, la version la plus récente de l'offre et **tous** les correctifs temporaires de cette version sont installés.
  - Si la *version_offre* est spécifiée, la version indiquée de l'offre est installée et **aucun** correctif temporaire de cette version n'est installé.

  La version de l'offre peut être rattachée à la fin de l'ID offre à l'aide d'un trait de soulignement

lorsque vous exécutez la commande suivante au niveau du référentiel :

```
imcl listAvailablePackages -repositories référentiel_source
```

- Vous pouvez également spécifier `none`, `recommended` ou `all` avec l'argument `-installFixes` afin d'indiquer les correctifs temporaires que vous souhaitez installer avec l'offre.
  - Si la version de l'offre n'est **pas** spécifiée, l'option `-installFixes` est paramétrée par défaut sur `all`.
  - Si la version de l'offre est spécifiée, l'option `-installFixes` est paramétrée par défaut sur `none`.
- Vous pouvez ajouter une liste de fonctions séparées par des virgules. Exemple de programme :

```
imcl -acceptLicense install
com.ibm.websphere.WXS.v85,xs.console.feature,xs.samples.feature
```

```
imcl -acceptLicense install
com.ibm.websphere.WXS.v86,xs.console.feature,xs.samples.feature
```

  - `xs.console.feature` Disponible pour toutes les offres de produit. Vous pouvez choisir d'installer la console de surveillance. Avec la console Web, vous pouvez générer des graphiques des statistiques actuelles et historiques. Cette console fournit un certain nombre de graphiques préconfigurés pour des présentations générales et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.
  - `xs.samples.feature` Disponible pour toutes les offres de produit. Vous pouvez choisir d'installer des exemples.

Remarques :

- Si vous avez précédemment spécifié le mode d'installation d'Installation Manager, le paramètre -accessRights n'est pas requis.
- Si vous rencontrez des problèmes ultérieurement, Installation Manager peut sauvegarder les versions antérieures d'un package à des fins de rétrogradation. Quand Installation Manager rétrograde un package à une version antérieure, la version actuelle des fichiers est désinstallée et les versions antérieures sont réinstallées. Si vous choisissez de ne pas sauvegarder les fichiers pour rétrogradation, vous pouvez empêcher la sauvegarde des fichiers en indiquant une préférence :

  ```
  -preference
  com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts=False
  ```

  Pour plus d'informations sur la définition de vos préférences pour Installation Manager, voir le centre de documentation d'IBM Installation Manager Version 1.5.

  Conseil : Même si vous choisissez de ne pas sauvegarder les fichiers pour rétrogradation, vous pouvez accéder aux fichiers de produit pour rétrogradation à partir du référentiel.

- Le programme peut générer des instructions post-installation importantes dans la sortie standard.

  Pour plus d'informations sur l'utilisation de la commande **`imcl`** pour installer le produit, voir le centre de documentation d'IBM Installation Manager Version 1.5.

Rubrique parent : Installation d'IBM Installation Manager à l'aide de la ligne de commande

Tâches associées:
Désinstallation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Java

Installation d'IBM Installation Manager à l'aide de fichiers de réponses

Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM® Installation Manager. Vous pouvez installer Installation Manager à l'aide de fichiers de réponses.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

Accédez à l'emplacement contenant les fichiers d'installation d'Installation Manager, puis exécutez une des commandes ci-dessous pour installer Installation Manager.

Installation par un administrateur :

- Windows `installc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
- UNIX Linux `./installc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

Installation par un non administrateur :

- Windows `userinstc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
- UNIX Linux `./userinstc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

Installation en mode groupe :

UNIX Linux `./groupinstc -acceptLicense -dataLocation` *`emplacement_données_application`* `-log` *`chemin_et_nom_fichier_journal`* `-installationDirectory` *`rép_Installation_Manager`*

Remarques sur le mode groupe :

- Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des packages logiciels.

  Le mode groupe ne signifie pas que deux personnes peuvent utiliser la même instance d'IBM Installation Manager en même temps.

- Windows Le mode groupe n'est pas disponible sur les systèmes d'exploitation Windows.
- Si vous n'installez pas Installation Manager avec le mode groupe, vous ne pourrez pas utiliser le mode groupe pour gérer les produits que vous installez ultérieurement à l'aide de cette instance d'Installation Manager.
- Assurez-vous de bien changer l'emplacement d'installation et de le faire passer de l'emplacement par défaut dans le répertoire de base de l'utilisateur actuel à un emplacement accessible par tous les utilisateurs du groupe.
- Configurez vos groupes, droits d'accès et variables d'environnement, comme décrit dans les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5 avant d'effectuer l'installation en mode groupe.
- Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

Que faire ensuite

Une fois que l'installation d'Installation Manager et la configuration du référentiel aboutissent, vous pouvez continuer d'installer WebSphere eXtreme Scale WebSphere eXtreme Scale Client ou WebSphere eXtreme Scale for WebSphere Application Server pour l'offre de produit. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.

Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses
Utilisez Installation Manager à l'aide d'un fichier de réponses pour installer les offres de produit WebSphere eXtreme Scale Client.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

Utilisez Installation Manager à l'aide d'un fichier de réponses pour installer les offres de produit WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Pourquoi et quand exécuter cette tâche

A l'aide d'Installation Manager, vous pouvez utiliser des fichiers de réponses pour installer le produit de plusieurs manières. Vous pouvez enregistrer un fichier de réponses à l'aide de l'interface graphique.

Procédure

1. A partir d'une ligne de commande, accédez au sous-répertoire Eclipse du répertoire dans lequel vous avez installé Installation Manager.
2. Démarrez Installation Manager à partir de la ligne de commande en utilisant l'option d'enregistrement -record.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     IBMIM.exe -skipInstall "C:\temp\imRegistry"
       -record C:\temp\install_response_file.xml
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./IBMIM -skipInstall /var/temp/imRegistry
       -record /var/temp/install_response_file.xml
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./IBMIM -skipInstall racine_utilisateur/var/temp/imRegistry
       -record rép_base_utilisateur/var/temp/install_response_file.xml
     ```

   Conseil : Lorsque vous enregistrez un nouveau fichier de réponses, vous pouvez sélectionner le paramètre -skipInstall. Celui-ci vous offrira les avantages suivants :

   - Aucun fichier n'est installé, ce qui accélère l'enregistrement.
   - Si vous utilisez un emplacement de données temporaire avec le paramètre -skipInstall, Installation Manager y placera le registre d'installation lors de l'enregistrement. Lorsque vous redémarrerez Installation Manager sans le paramètre -skipInstall, vous pourrez utiliser le fichier de réponses pour l'installation au lieu du vrai registre d'installation.

     L'opération -skipInstall ne doit pas être utilisée sur l'emplacement de données de l'agent en cours utilisé par Installation Manager. Cette opération n'est pas prise en charge. Utilisez un nouvel emplacement inscriptible et utilisez à nouveau l'emplacement pour des sessions d'enregistrement ultérieures.

   Pour plus d'informations, voir le centre de documentation d'IBM Installation Manager Version 1.5.
3. Ajoutez les référentiels appropriés à vos préférences dans Installation Manager.
   a. Dans le menu du haut, cliquez sur **File > Preferences**.
   b. Sélectionnez **Repositories**
   c. Effectuez les actions suivantes pour chaque référentiel :
      i. Cliquez sur **Add Repository**.
      ii.
      iii. Entrez le chemin d'accès au fichier `repository.config` du référentiel Web éloigné ou du répertoire local dans lequel vous avez décompressé les fichiers du référentiel.

           Par exemple :

- Référentiels éloignés :

  ```
  https://downloads.mycorp.com:8080/WXS_85_repository
  ```

- Référentiels locaux :
  - Windows `C:\repositories\wxs85\local-repositories`
  - UNIX Linux `/var/repositories/wxs85/local-repositories`

iv. Cliquez sur **OK**.
v. Cliquez sur **Appliquer**.
vi. Cliquez sur **OK**.

d. Cliquez sur **Install**.

Remarque : Si vous êtes invité à vous authentifier, utilisez l'ID et le mot de passe IBM avec lesquels vous vous êtes enregistré sur le site Web du programme.

Installation Manager analyse ses référentiels définis à la recherche des packages disponibles.

4. Sélectionnez l'une des offres de produit suivantes ainsi que la version appropriée :
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

   Si le produit est déjà installé sur votre système, un message s'affichera pour vous en informer. Pour créer une autre installation du produit dans un autre emplacement, cliquez sur **Continuer**.

   Conseil : Si l'option **Search service repositories during installation and updates** est sélectionnée sur la page des préférences de référentiel dans Installation Manager et que vous êtes connecté à Internet, vous pouvez cliquer sur **Check for Other Versions and Extensions**. Ainsi, vous pouvez rechercher les mises à jour dans les référentiels de mise à jour par défaut pour les packages sélectionnés. Dans ce cas, il n'est pas nécessaire d'ajouter l'URL de référentiel de service spécifique à la page des préférences en matière de référentiel d'Installation Manager.

5. Sélectionnez les correctifs à installer.

   Les correctifs recommandés sont sélectionnés par défaut.

   Vous pouvez choisir de n'afficher que les correctifs recommandés et de masquer les autres.

6. Cliquez sur **Suivant**.
7. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
8. Indiquez le répertoire principal d'installation du produit.

   Le panneau affiche également le répertoire de ressources partagées et les informations d'espace disque.

   Remarque : La première fois que vous installez un package à l'aide d'Installation Manager, spécifiez le répertoire des ressources partagées. Le répertoire des ressources partagées est l'endroit où les artefacts d'installation se trouvent ; ils peuvent être utilisés par un plusieurs groupe de packages. Utilisez votre unité ayant la taille la plus importante pour cette installation. Vous ne pouvez pas changer l'emplacement du répertoire tant que vous n'avez pas désinstallé tous les packages.

   Restrictions :

   - Si vous supprimez l'emplacement cible par défaut et que vous ne renseignez pas une zone du répertoire d'installation, vous ne pouvez pas continuer.
   - N'utilisez pas de liens symboliques comme répertoire de destination.

     car ils ne sont pas pris en charge.
   - Windows La longueur maximale du chemin d'accès sur les systèmes d'exploitation Windows Server 2008, Windows Vista et Windows 7 est de 60 caractères.

9. Cliquez sur **Suivant**.
10. Sélectionnez les langues pour lesquelles un contenu traduit doit être installé.

    L'anglais est toujours sélectionné.

11. Cliquez sur **Suivant**.

12. Sélectionnez les fonctions que vous voulez installer.
    - Console

      Disponible pour toutes les offres de produit WebSphere eXtreme Scale. Vous pouvez choisir d'installer la console de surveillance. Avec la console Web, vous pouvez générer des graphiques des statistiques actuelles et historiques. Cette console fournit un certain nombre de graphiques pour des présentations générales et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.

    - Samples

      Disponible pour toutes les offres de produit WebSphere eXtreme Scale.

13. Cliquez sur **Suivant**.
14. Lisez le récapitulatif et cliquez sur **Install**.
    - Si l'installation aboutit, le programme affiche un message indiquant que l'installation a abouti.

      Remarque : Le programme peut également afficher d'importantes instructions post-installation.

    - Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
15. Cliquez sur **Finish**.
16. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Création d'un fichier de clés
Après avoir utilisé Installation Manager pour enregistrer un fichier de réponses afin d'installer des offres de produit WebSphere eXtreme Scale WebSphere eXtreme Scale Client, vous pouvez choisir de créer un fichier de clés. Si vous utilisez un référentiel distant nécessitant une authentification, vous pouvez créer un fichier clés pour l'installation.

Rubrique parent : Installation d'IBM Installation Manager à l'aide de fichiers de réponses

Tâches associées:
Désinstallation de WebSphere eXtreme Scale Client à l'aide de fichiers de réponses

Java

Création d'un fichier de clés

Après avoir utilisé Installation Manager pour enregistrer un fichier de réponses afin d'installer des offres de produit WebSphere eXtreme Scale WebSphere eXtreme Scale Client, vous pouvez choisir de créer un fichier de clés. Si vous utilisez un référentiel distant nécessitant une authentification, vous pouvez créer un fichier clés pour l'installation.

Avant de commencer

Vous devez enregistrer un fichier de réponses. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses.

Procédure

1. A partir d'une ligne de commande, accédez au sous-répertoire Eclipse du répertoire dans lequel vous avez installé Installation Manager.
2. Démarrez Installation Manager à partir de la ligne de commande en utilisant l'option d'enregistrement -record.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     IBMIM.exe -skipInstall "C:\temp\imRegistry"
       -keyring C:\IM\im.keyring
       -record C:\temp\keyring_response_file.xml
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./IBMIM -skipInstall /var/temp/imRegistry
       -keyring /var/IM/im.keyring
       -record /var/temp/keyring_response_file.xml
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./IBMIM -skipInstall racine_utilisateur/var/temp/imRegistry
       -keyring rép_base_utilisateur/var/IM/im.keyring
       -record racine_utilisateur/var/temp/keyring_response_file.xml
     ```

3. Lorsqu'une fenêtre s'affiche, vous demandant vos données d'identification pour le référentiel distant authentifié, entrez-les et **enregistrez**-les.
4. Cliquez sur **File > Exit** pour fermer Installation Manager.

   Pour plus d'informations, voir le centre de documentation d'IBM® Installation Manager Version 1.5.

Rubrique parent : Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Utilisez IBM® Installation Manager pour désinstaller des offres de produit WebSphere eXtreme Scale Client.

Désinstallation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique
Vous pouvez utiliser la console de l'assistant d'IBM Installation Manager pour désinstaller WebSphere eXtreme Scale Client.

Désinstallation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande
Vous pouvez désinstaller WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager à partir de la ligne de commande.

Désinstallation de WebSphere eXtreme Scale Client à l'aide de fichiers de réponses
Vous pouvez désinstaller WebSphere eXtreme Scale Client avec IBM Installation Manager à l'aide de fichiers de réponses.

Rubrique parent : Installation de WebSphere eXtreme Scale Client

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Vous pouvez utiliser la console de l'assistant d'IBM® Installation Manager pour désinstaller WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez supprimer l'argument WebSphere eXtreme Scale de tous les profils WebSphere Application Server avant de désinstaller WebSphere eXtreme Scale. Vous ne pourrez pas exécuter l'annulation d'extension après avoir désinstallé WebSphere eXtreme Scale. Utilisez la commande manageprofiles pour réduire les profils existants dans un environnement WebSphere eXtreme Scale. .

Procédure

1. Désinstallez le produit.
    a. Arrêtez tous les processus en cours d'exécution dans votre environnement.
        - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
    b. Démarrez Installation Manager.
    c. Cliquez sur **Désinstaller**.
    d. Dans la fenêtre **Désinstaller des packages**, effectuez les actions suivantes.
        i. Sélectionnez l'une des offres de produit suivantes ainsi que la version appropriée :
            - WebSphere eXtreme Scale Client dans un environnement autonome
            - WebSphere eXtreme Scale Client for WebSphere Application Server Version 6
            - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
            - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
        ii. Cliquez sur **Next**.
    e. Si l'assistant de désinstallation affiche la liste des profils étendus, WebSphere Application Server, vous devez annuler l'extension de ces profils pour pouvoir poursuivre la désinstallation.
    f. Examinez les informations récapitulatives.
    g. Cliquez sur **Désinstaller**.
        - Si la désinstallation a réussi, le programme affichera un message pour confirmer le succès de l'opération.
        - Dans le cas contraire, cliquez sur **View log** pour corriger l'erreur.
    h. Cliquez sur **Terminer**.
    i. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.
2. Facultatif : Désinstallez IBM Installation Manager.

    Important : Avant de pouvoir désinstaller IBM Installation Manager, vous devez désinstaller tous les packages installés par Installation Manager.

    Pour plus d'informations sur cette procédure, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

Rubrique parent : Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Vous pouvez désinstaller WebSphere eXtreme Scale Client à l'aide d'IBM® Installation Manager à partir de la ligne de commande.

Avant de commencer

Vous devez supprimer l'argument WebSphere eXtreme Scale de tous les profils WebSphere Application Server avant de désinstaller WebSphere eXtreme Scale. Vous ne pourrez pas exécuter l'annulation d'extension après avoir désinstallé WebSphere eXtreme Scale. Utilisez la commande manageprofiles pour réduire les profils existants dans un environnement WebSphere eXtreme Scale.

Procédure

1. Ouvrez une session sur votre système.
2. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
3. Accédez au sous-répertoire `eclipse/tools` dans le répertoire où vous avez installé Installation Manager.
4. Utilisez la commande **`imcl`** pour désinstaller le produit.

   Windows

   ```
   imcl.exe uninstall com.ibm.websphere.v85,ID_fonction_facultative
     -installationDirectory répertoire_installation
   ```

   UNIX Linux

   ```
   ./imcl uninstall com.ibm.websphere.v85,ID_fonction_facultative
     -installationDirectory répertoire_installation
   ```

   Conseils :
   - *`ID_offre`* correspond à l'ID offre répertorié dans la rubrique ID d'offre pour les produits WebSphere eXtreme Scale Client.
   - Vous pouvez supprimer une liste de fonctions séparées par des virgules (ID fonction). Par exemple,

     ```
     imcl uninstall com.ibm.websphere.WXS.v85,xs.console.feature,xs.samples.feature
     ```

     - `client` indique la fonction client autonome
     - `server` indique la fonction serveur autonome
     - `console` indique la console de surveillance Web
     - `samples` indique les exemples
   - Si aucune liste de fonctions n'est spécifiée, la totalité du produit est désinstallée.

   Pour plus d'informations, accédez au d'centre de documentation d'IBM Installation Manager Version 1.5.
5. Si le processus de désinstallation affiche la liste des profils WebSphere Application Server étendus, vous devez annuler l'extension de ces profils pour pouvoir poursuivre la désinstallation.
6. Facultatif : Désinstallez IBM Installation Manager.

   Important : Avant de pouvoir désinstaller IBM Installation Manager, vous devez désinstaller tous les packages installés par Installation Manager.

   Pour plus d'informations sur l'utilisation du script de désinstallation pour exécuter cette procédure, consultez le centre de documentation d'IBM Installation Manager version 1.5.

Rubrique parent : Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide de fichiers de réponses

Vous pouvez désinstaller WebSphere eXtreme Scale Client avec IBM® Installation Manager à l'aide de fichiers de réponses.

Avant de commencer

Vous devez supprimer l'argument WebSphere eXtreme Scale de tous les profils WebSphere Application Server avant de désinstaller WebSphere eXtreme Scale. Vous ne pourrez pas exécuter l'annulation d'extension après avoir désinstallé WebSphere eXtreme Scale. Utilisez la commande manageprofiles pour réduire les profils existants dans un environnement WebSphere eXtreme Scale.

Facultatif : Effectuez ou enregistrez l'installation d'Installation Manager et l'installation du produit dans un registre d'installation temporaire sur l'un de vos systèmes de manière à pouvoir enregistrer la désinstallation dans ce registre temporaire et non dans le registre standard ou est installé Installation Manager.

Pourquoi et quand exécuter cette tâche

Avec Installation Manager, vous pouvez utiliser des fichiers de réponses pour désinstaller le produit de plusieurs manières. Vous pouvez enregistrer un fichier de réponses à l'aide de l'interface graphique tel que décrit dans la procédure suivante, ou bien vous pouvez générer un nouveau fichier de réponses à la main ou en modifiant un exemple .

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
2. Facultatif : **Enregistrez un fichier de réponse pour désinstaller le produit :** Sur l'un de vos systèmes, effectuez les actions suivantes pour enregistrer un fichier de réponse qui vous permettra de désinstaller le module de fonctions:
   a. A partir d'une ligne de commande, passez au sous-répertoire eclipse du répertoire dans lequel vous avez installé Installation Manager.
   b. Démarrez Installation Manager à partir de la ligne de commande en utilisant l'option d'enregistrement -record.

      Par exemple :

      - Windows **Administrateur ou non administrateur :**

        ```
        IBMIM.exe -skipInstall "C:\temp\imRegistry"
          -record C:\temp\uninstall_response_file.xml
        ```

      - UNIX Linux **Administrateur :**

        ```
        ./IBMIM -skipInstall /var/temp/imRegistry
          -record /var/temp/uninstall_response_file.xml
        ```

      - UNIX Linux **Non administrateur :**

        ```
        ./IBMIM -skipInstall racine_utilisateur/var/temp/imRegistry
          -record racine_utilisateur/var/temp/uninstall_response_file.xml
        ```

Conseil : Si vous choisissez d'utiliser le paramètre -skipInstall avec un registre d'installation temporaire créé en suivant les instructions *Avant de commencer*, Installation Manager utilise le registre d'installation temporaire lors de l'enregistrement du fichier de réponses. Il est important de savoir que dès lors que le paramètre -skipInstall est spécifié, aucun package de fonction n'est installé ou désinstallé. Toutes les actions effectuées dans Installation Manager mettent tout simplement à jour les données d'installation stockées dans le registre temporaire indiqué. Une fois que le fichier de réponses est généré, il peut être utilisé pour désinstaller le produit : il supprime les fichiers du produit et met à jour le registre d'installation standard.

L'opération -skipInstall ne doit pas être utilisée sur l'emplacement de données de l'agent en cours utilisé par Installation Manager. Cette opération n'est pas prise en charge. Utilisez un nouvel emplacement inscriptible et utilisez à nouveau l'emplacement pour des sessions d'enregistrement ultérieures.

Pour plus d'informations, voir le centre de documentation d'IBM Installation Manager Version 1.5.

c. Cliquez sur **Désinstaller**.
d. Dans la fenêtre **Désinstaller des packages**, effectuez les actions suivantes.
    i. Sélectionnez l'une des offres de produit suivantes ainsi que la version appropriée :
        - WebSphere eXtreme Scale Client dans un environnement autonome
        - WebSphere eXtreme Scale Client for WebSphere Application Server Version 6
        - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
        - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
    ii. Cliquez sur **Next**.
    iii. Cliquez sur **Suivant**.
e. Examinez les informations récapitulatives.
f. Cliquez sur **Désinstaller**.
    - Si la désinstallation a réussi, le programme affichera un message pour confirmer le succès de l'opération.
    - Dans le cas contraire, cliquez sur **View log** pour corriger l'erreur.
g. Cliquez sur **Terminer**.
h. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

3. **Utilisez le fichier de réponses pour désinstaller le produit :** A partir d'une ligne de commande sur chacun des systèmes desquels vous souhaitez désinstaller le produit, accédez au sous-répertoire `eclipse/tools` du répertoire dans lequel vous avez installé Installation Manager et utilisez le fichier de réponses que vous avez créé pour procéder à la désinstallation du produit.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     imcl.exe
       input C:\temp\uninstall_response_file.xml
       -log C:\temp\uninstall_log.xml
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./imcl
       input /var/temp/uninstall_response_file.xml
       -log /var/temp/uninstall_log.xml
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./imcl
       input rép_base_utilisateur/var/temp/uninstall_response_file.xml
       -log racine_utilisateur/var/temp/uninstall_log.xml
     ```

   Pour plus d'informations, voir le centre de documentation d'IBM Installation Manager Version 1.5.

4. Facultatif : Répertoriez tous les packages installés pour vérifier la désinstallation.

   UNIX Linux

   ```
   ./imcl listInstalledPackages
   ```

   Windows

   ```
   imcl listInstalledPackages
   ```

5. Si le processus de désinstallation affiche la liste des profils WebSphere Application Server étendus, vous devez annuler l'extension de ces profils pour pouvoir poursuivre la désinstallation.
6. Facultatif : Désinstallez IBM Installation Manager.

   Important : Avant de pouvoir désinstaller IBM Installation Manager, vous devez désinstaller tous les packages installés par Installation Manager.

Pour plus d'informations sur l'utilisation du script de désinstallation pour exécuter cette procédure, accédez au centre de documentation d'IBM Installation Manager version 1.5.

Rubrique parent : Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

.NET

Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Vous pouvez installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution ou dans un environnement d'exécution et de production.

Pour générer et tester vos propres applications .NET, installez WebSphere eXtreme Scale Client for .NET dans votre environnement de développement. L'installation de l'environnement de développement inclut toujours l'installation de l'environnement d'exécution. Les assemblages d'exécution sont installés sur le disque et dans le cache GAC (Global Assembly Cache). L'installation de l'environnement de développement installe en outre du code exemple, l'intégration Visual Studio IntelliSense (pour les descriptions de classe et de méthode fly-over) et une documentation d'API. L'exemple de code source WebSphere eXtreme Scale Client for .NET et le projet Visual Studio sont installés dans le répertoire *net_client_home*\sample, et la documentation d'API est installée dans le répertoire *net_client_home*\doc.

.NET 2.5+ **Installation de WebSphere eXtreme Scale Client for .NET**
Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET si vous possédez des applications qui s'exécutent dans cette structure.

.NET 2.5+ **Installation de WebSphere eXtreme Scale Client for .NET en mode silencieux**
Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET en mode silencieux si vous ne souhaitez pas afficher le déroulement de l'installation ou devez installer le produit sur plusieurs machines. L'installation en mode silencieux signifie que vous devez enregistrer d'abord un fichier de réponses et les paramètres transmis à ce fichier.

.NET 2.5+ **Désinstallation de WebSphere eXtreme Scale Client for .NET**
Pour supprimer WebSphere eXtreme Scale Client for .NET dans votre environnement, vous pouvez le désinstaller depuis le panneau de configuration Windows ou enregistrer un fichier de réponses pour le désinstaller en mode silencieux. Il est préférable d'enregistrer un fichier de réponses lorsque vous avez plusieurs installations de WebSphere eXtreme Scale Client et que vous voulez supprimer rapidement ces installations.

.NET 2.5+ **Comment installer WebSphere eXtreme Scale Client for .NET sans utiliser le programme d'installation**
Si vous n'êtes pas autorisé à exécuter le fichier setup.exe, vous pouvez copier les fichiers d'une installation existante sur un autre système Windows.

Rubrique parent : Installation de WebSphere eXtreme Scale Client

.NET

Installation de WebSphere eXtreme Scale Client for .NET

Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET si vous possédez des applications qui s'exécutent dans cette structure.

Avant de commencer

- Installez WebSphere eXtreme Scale Client for .NET à partir du DVD. Le fichier `setup.exe` se trouve dans le répertoire racine, `/ClientForDotNet/setup.exe` (vous pouvez aussi le télécharger à partir du Site de support).
- Si vous comptez installer WebSphere eXtreme Scale Client for .NET dans un environnement de développement, vous devez utiliser un système Windows qui respecte la configuration matérielle et logicielle requise. Pour plus d'informations, voir Remarques relatives à Microsoft .NET.

Pourquoi et quand exécuter cette tâche

Vous pouvez installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution uniquement, ou dans un environnement d'exécution et dans un environnement de développement.

Procédure

1. Utilisez l'assistant pour installer le client à partir du DVD. Le fichier `setup.exe` se trouve dans le répertoire racine, `/ClientForDotNet/setup.exe`, ou vous pouvez le télécharger à partir du Site de support.
2. Exécutez le fichier `setup.exe`.
3. Suivez les invites de l'assistant et cliquez sur **Suivant** pour accéder à la page du type de configuration.
4. Si vous avez décidé d'installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution, cliquez sur **Environnement d'exécution**, puis procédez comme suit :

   Cliquez sur **Installer** pour exécuter le programme d'installation, puis cliquez sur **Terminer**. Le répertoire d'installation par défaut est `C:\Program Files (x86)\IBM\WebSphere\eXtreme Scale .NET Client`
5. Si vous avez décidé d'installer WebSphere eXtreme Scale Client dans un environnement d'exécution et dans un environnement de développement, choisissez l'installation **Personnalisée** et procédez comme suit :
   a. Installez WebSphere eXtreme Scale Client dans le répertoire d'installation par défaut ou choisissez votre propre répertoire d'installation et cliquez sur **Suivant**.
   b. Par défaut, l'environnement d'exécution et l'environnement de développement sont sélectionnés. Vérifiez que vous disposez d'un espace disque suffisant pour installer les deux environnements. Cliquez sur **Suivant**.
   c. Choisissez l'emplacement des fichiers journaux et cliquez sur **Suivant**.
   d. Cliquez sur **Installer** pour exécuter le programme d'installation, puis cliquez sur **Terminer**.

Que faire ensuite

- Testez votre WebSphere eXtreme Scale Client for .NET en exécutant l'application d'initiation. Pour plus d'informations, voir Tutoriel : Démarrer avec des applications de grille de données simples.
- Configurez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de WebSphere eXtreme Scale Client for .NET.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
2.5+ Désinstallation de WebSphere eXtreme Scale Client for .NET
.NET 2.5+ Comment installer WebSphere eXtreme Scale Client for .NET sans utiliser le programme d'installation

.NET

Installation de WebSphere eXtreme Scale Client for .NET en mode silencieux

Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET en mode silencieux si vous ne souhaitez pas afficher le déroulement de l'installation ou devez installer le produit sur plusieurs machines. L'installation en mode silencieux signifie que vous devez enregistrer d'abord un fichier de réponses et les paramètres transmis à ce fichier.

Avant de commencer

- Procurez-vous WebSphere eXtreme Scale Client sur DVD. Le fichier `setup.exe` se trouve dans le répertoire racine, `/net_client_home/setup.exe`, ou vous pouvez le télécharger à partir du Site de support.
- Si vous envisagez d'installer WebSphere eXtreme Scale Client for .NET dans un environnement de développement, vous devez utiliser un système Windows qui respecte la configuration matérielle et logicielle décrite dans les remarques relatives à Microsoft .NET. Pour plus d'informations, voir Remarques relatives à Microsoft .NET.

Pourquoi et quand exécuter cette tâche

WebSphere eXtreme Scale Client for .NET peut être installé dans un environnement d'exécution ou dans un environnement d'exécution et de développement.

Procédure

1. Ouvrez une invite de commande et exécutez le script suivant : `setup.exe /r /f1"<Response_Files_Directory>\Setup.iss"` `<Response_Files_Directory>` est l'emplacement où vous voulez créer le fichier de réponses.
2. Suivez les invites de l'assistant et cliquez sur **Suivant** pour accéder à la page du type de configuration.
3. En fonction des options que vous choisissez, vous pouvez transmettre les valeurs suivantes pour créer le fichier de réponses `Setup.iss` :
   - Choisissez d'installer WebSphere eXtreme Scale Client dans un environnement d'exécution ou choisissez une installation personnalisée. Une personnalisation personnalisée permet d'installer le produit dans les deux environnements.
   - Si vous avez décidé d'installer WebSphere eXtreme Scale Client dans un environnement d'exécution, cliquez sur **Environnement d'exécution** et procédez comme suit :
     a. Cliquez sur **Désinstaller** et cliquez sur **Terminer**. L'installation par défaut est `C:\Program Files (x86)\IBM\WebSphere\eXtreme Scale .NET Client`
   - Si vous avez décidé d'installer WebSphere eXtreme Scale Client dans un environnement et un environnement d'exécution, choisissez l'installation **personnalisée** et procédez comme suit :
     a. Installez WebSphere eXtreme Scale Client dans le répertoire d'installation par défaut ou choisissez votre propre répertoire d'installation. Cliquez sur **Suivant**.
     b. Sélectionnez un environnement d'exécution et un environnement de développement. Si vous souhaitez effectuer l'installation dans les deux environnements, vérifiez que vous disposez d'un espace disque suffisant. Cliquez sur **Suivant**
     c. Choisissez l'emplacement des fichiers journaux et cliquez sur **Suivant**.
4. Cliquez sur **Désinstaller** et cliquez sur **Terminer**.
5. Ouvrez une invite de commande et exécutez le script suivant pour installer WebSphere eXtreme Scale Client en mode silencieux : `setup.exe /s /f1"<Response_Files_Directory>\Setup.iss"`, où `<Response_Files_Directory>` est l'emplacement où réside votre fichier de réponses.

Que faire ensuite

Vous pouvez mettre à jour ou modifier `SimpleClient` pour tester les API du client eXtreme Scale. Recherchez `SimpleClient` dans `<installation_directory>\sample\SimpleClient` dans le répertoire d'installation, puis chargez ce fichier dans Visual Studio afin d'afficher l'exemple d'application qui utilise des opérations simples de création, d'extraction, de mise à jour et de suppression. Utilisez `SimpleClient` comme guide pour accéder à la grille de données. Vous pouvez modifier cette application ou écrire une nouvelle application qui utilise le groupe d'API prises en charge du client eXtreme Scale for .NET. Pour plus d'informations, voir Tutoriel : Démarrer avec des applications de grille de données simples.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
2.5+ Désinstallation de WebSphere eXtreme Scale Client for .NET

.NET

Désinstallation de WebSphere eXtreme Scale Client for .NET

Pour supprimer WebSphere eXtreme Scale Client for .NET dans votre environnement, vous pouvez le désinstaller depuis le panneau de configuration Windows ou enregistrer un fichier de réponses pour le désinstaller en mode silencieux. Il est préférable d'enregistrer un fichier de réponses lorsque vous avez plusieurs installations de WebSphere eXtreme Scale Client et que vous voulez supprimer rapidement ces installations.

Avant de commencer

Si vous souhaitez désinstaller le produit dans un environnement de développement, veillez à arrêter Visual Studio.

Avertissement : Le programme de désinstallation supprime tous les fichiers binaires et la maintenance, telle que les groupes de correctifs et les correctifs temporaires, en même temps.

Procédure

1. Arrêtez les processus .NET eXtreme Scale.
2. Vous pouvez désinstaller WebSphere eXtreme Scale Client for .NET au moyen de l'une des méthodes suivantes :
   - Effectuez la désinstallation à partir du panneau de configuration Windows, cliquez sur Ajouter ou supprimer des programmes, puis sélectionnez IBM WebSphere eXtreme Scale Client for .NET.
   - Si vous voulez enregistrer un fichier de réponses, ouvrez une invite de commande et exécutez le script suivant :

     ```
     setup.exe /uninst /r /f1"<Response_Files_Directory>\Setup.iss"
     ```

     a. L'assistant de désinstallation s'ouvre et une fenêtre de confirmation s'affiche pour vérifier que vous voulez supprimer WebSphere eXtreme Scale Client for .NET et toutes ses fonctions. Cliquez sur **OK**.
     b. A la fin de la désinstallation, cliquez sur **Terminer**.
3. Facultatif : si vous voulez utiliser votre fichier de réponses pour désinstaller une installation existante de WebSphere eXtreme Scale Client, exécutez votre fichier de réponses comme suit :

   a. Ouvrez une invite de commande et exécutez le script suivant pour désinstaller WebSphere eXtreme Scale Client for .NET en mode silencieux :

   ```
   setup.exe /uninst /s /f1"<Response_Files_Directory>\Setup.iss"
   ```

Que faire ensuite

Vérifiez Windows Explorer pour vous assurer que tous les dossiers ont été supprimés du répertoire d'installation. Vous devez également vérifier le panneau de configuration Windows pour vous assurer que le produit n'est plus répertorié. Le programme d'installation ne supprime pas les dossiers comportant des fichiers qui ont été générées après l'installation, par exemple les fichiers journaux, les fichiers de configuration personnalisés et les artefacts créés par la génération de l'exemple `SimpleClient`.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET 2.5+ Installation de WebSphere eXtreme Scale Client for .NET
.NET 2.5+ Installation de WebSphere eXtreme Scale Client for .NET en mode silencieux

.NET

Comment installer WebSphere eXtreme Scale Client for .NET sans utiliser le programme d'installation

2.5+ Si vous n'êtes pas autorisé à exécuter le fichier `setup.exe`, vous pouvez copier les fichiers d'une installation existante sur un autre système Windows.

Avant de commencer

Vous devez disposer d'une installation opérationnelle de WebSphere eXtreme Scale Client for .NET à partir de laquelle vous pourrez copier les fichiers. Pour plus d'informations, voir Présentation de l'installation de WebSphere eXtreme Scale Client for .NET.

Pourquoi et quand exécuter cette tâche

Si vous n'êtes pas autorisé à exécuter le fichier `setup.exe` sur votre serveur client, vous pouvez copier les fichiers d'une installation existante de WebSphere eXtreme Scale Client for .NET dans le répertoire d'exécution de votre application .NET.

Avertissement : Les fonctionnalités de migration et de mise à niveau automatiques ne sont disponibles que dans le programme d'installation de WebSphere eXtreme Scale Client for .NET. Si vous utilisez la procédure ci-dessous pour installer le produit, vous devrez donc effectuer manuellement les migrations et les mises à niveau.

Procédure

1. Windows A partir de votre installation de WebSphere eXtreme Scale Client for .NET, copiez les fichiers ci-dessous sur le système Windows cible. Le répertoire dans lequel vous effectuez la copie sur le système Windows (répertoire cible) doit être le répertoire d'exécution du processus qui exécute l'application .NET pour votre grille de données. Ce processus doit disposer d'un accès en lecture et en écriture au répertoire cible. Les fichiers journaux sont créés dans un dossier `logs` dans ce répertoire.
   - *net_client_home*`\bin\IBM.WebSphere.Caching.dll`
   - *net_client_home*`\bin\IBM.WebSphere.Caching.CredentialGenerator.dll` (ce fichier ne doit être copié que si l'application WebSphere eXtreme Scale Client for .NET est configurée pour utiliser le générateur de données d'identification fourni)
   - *net_client_home*`\config\Client.Net.properties`
   - *net_client_home*`\config\Client.Net.Log.config`
2. Facultatif : Installez manuellement l'assemblage `IBM.WebSphere.Caching.dll` dans le cache GAC (Global Assembly Cache). La procédure à suivre dépend de votre environnement Windows.

Que faire ensuite

Configurez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de WebSphere eXtreme Scale Client for .NET.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET 2.5+ Installation de WebSphere eXtreme Scale Client for .NET

Java

Installation de profil Liberty

Vous installez l'environnement de service d'applications profil Liberty en utilisant le gestionnaire d'installation ou en exécutant un fichier d'archive Java™ (fichier JAR).

Pourquoi et quand exécuter cette tâche

Pour installer le profil Liberty, vous pouvez installer le WebSphere eXtreme Scale Client pour WebSphere Application Server version 8.5 ou ultérieur, ou exécuter un fichier JAR fourni..

Sur les systèmes z/OS, le profil Liberty fournit un environnement d'exécution. Vous pouvez utiliser cet environnement en mode natif à l'aide de la console MVS. Pour le développement d'applications, vous pouvez envisager d'utiliser les outils basés sur la technologie Eclipse sur un système réparti distinct, un système d'exploitation Mac OS, ou dans un shell Linux sur un système z/OS.

Procédure

- Installez le profil Liberty de WebSphere Application Server de l'une des manières suivantes :
  - Dans WebSphere DataPower XC10 Appliance, vous devez installer le WebSphere eXtreme Scale Client. Pour pouvoir exécuter le profil Liberty dans cette topologie, vous devez Installer l'environnement de service d'applications profil Liberty en exécutant un fichier d'archive Java (fichier JAR).
- Installez WebSphere eXtreme Scale Client version 8.5 ou ultérieur. Lorsque vous installez le produit de cette manière, le profil Liberty est automatiquement installé.
- Faites migrer votre environnement de profil Liberty. Lorsque vous migrez d'une édition majeure du profil Liberty vers une édition majeure ultérieure, vous devez modifier les numéros de version de fonctionnalité dans le fichier `server.xml` de votre profil Liberty.

  Par exemple, dans la version initiale du profil Liberty prise en charge par le produit, les numéros de version de fonctionnalité étaient au niveau `1.0`. Dans WebSphere eXtreme Scale Client version 8.6 et ultérieur, ces numéros sont au niveau `1.1`.

  Remarque : Lorsque vous effectuez une mise à niveau vers WebSphere Application Server version 8.5.5, le profil Liberty est supprimé. WebSphere eXtreme Scale prend en charge la suppression de cette fonctionnalité. Cependant, si vous revenez ensuite à WebSphere Application Server version 8.5, le profil Liberty est de nouveau ajouté, ce qui crée des problèmes dans WebSphere eXtreme Scale.

Profil Liberty et mise en cache des données
Vous pouvez utiliser des produits de mise en cache des données avec le profil Liberty de WebSphere Application Server afin de développer des sessions HTTP ainsi que des connexions client-serveur utilisant la passerelle REST, et de gérer d'autres scénarios d'intégration de cache.

Installation de l'environnement de service d'applications profil Liberty en exécutant un fichier JAR
En exécutant le fichier d'archive Java (JAR) qui contient l'image de distribution, vous installez l'environnement de service d'applications et vous êtes prêt à créer un serveur Liberty.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique
Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande
Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

Profil Liberty et mise en cache des données

Java Vous pouvez utiliser des produits de mise en cache des données avec le profil Liberty de WebSphere Application Server afin de développer des sessions HTTP ainsi que des connexions client-serveur utilisant la passerelle REST, et de gérer d'autres scénarios d'intégration de cache.

Dans WebSphere DataPower XC10 Appliance, vous pouvez utiliser le profil Liberty pour établir une connexion à la grille de données du dispositif. Par exemple, lorsque vous installez WebSphere eXtreme Scale Client avec le profil Liberty, vous avez accès à des fonctions qui vous permettent de gérer les applications de session HTTP, les applications client Java et les applications client REST installées dans le profil Liberty.

Les fonctions ci-dessous contiennent des informations concernant les principales fonctions disponibles. L'inclusion d'une fonction dans la configuration peut entraîner le chargement automatique d'une ou plusieurs fonctions. Chaque fonction comporte une brève description et un exemple de la façon dont cette fonction est déclarée.

Fonction client

La fonction client contient la plus grande partie du modèle de programmation pour eXtreme Scale. Ajoutez la fonction client lorsqu'une application qui s'exécute dans le profil Liberty utilisera des API eXtreme Scale.

Fichier *racine_installation_wlp*/usr/server/*racine_installation_wlp*/server.xml

```
<server description="WebSphere eXtreme Scale Client">

  <featureManager>
    <feature>eXtremeScale.client-1.1</feature>
  </featureManager>

 </server>
```

Fonction Web

La fonction Web est obsolète. Utilisez la fonction webApp pour répliquer les données de session HTTP pour la tolérance aux pannes.

La fonction Web permet d'étendre le profil Liberty à une application Web. Ajoutez la fonction Web lorsque vous souhaitez répliquer des données de session HTTP dans le cadre de la tolérance aux pannes.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

  <featureManager>
    <feature>eXtremeScale.web-1.1</feature>
  </featureManager>

  <xsWebAppV85/>
</server>
```

Fonction webApp

La fonction webApp permet d'étendre l'application Web du profil Liberty. Ajoutez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.webApp-1.1</feature>
</featureManager>

<xsWebApp/>
</server>
```

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">
```

```
<featureManager>
<feature>eXtremeScale.webGrid-1.1</feature>
</featureManager>

<xsWebGrid/>
</server>
```

Fonction REST

Utilisez la passerelle REST (Representational State Transfer) pour accéder aux grilles de données simples hébergées par un collectif dans le profil profil Liberty.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.rest-1.1</feature>
</featureManager>

<xsRest/>
</server>
```

Fonction de cache dynamique

Un serveur profil Liberty peut héberger une grille de données qui met en mémoire cache les données des applications pour lesquelles la mémoire cache dynamique est activée.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

  <featureManager>
    <feature>eXtremeScale.dynacacheGrid-1.1</feature>
  </featureManager>

  <xsDynacacheGrid/>
</server>
```

2.5+

Fonction de cache dynamique

Le serveur de profil Liberty peut héberger WebSphere eXtreme Scale, que vous pouvez configurer comme fournisseur de cache dynamique à l'aide de cette fonction.

Voyez l'exemple suivant pour savoir comment utiliser le fournisseur de cache par défaut dans le profil Liberty:

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.dynacacheapp-1.1</feature>
</featureManager>

<xsClientDomain default="production">
<endpointConfig> production;localhost:2809 </endpointConfig>
</xsClientDomain>

<distributedMap id="baseCache" libraryRef="idgenerator" cacheProviderName="WebSphere
eXtreme Scale">
<xsDynacacheApp remoteDomain="production" />
</distributedMap>

<distributedMap id="cache01" jndiName="cache01" memorySizeInEntries="2000"
createCacheAtServerStartup="true" cacheProviderName="WebSphere eXtreme Scale">
<xsDynacacheApp remoteDomain="production" />
```

```
</distributedMap>
</server>
```

Rubrique parent : Installation de profil Liberty

Java

Installation de l'environnement de service d'applications profil Liberty en exécutant un fichier JAR

En exécutant le fichier d'archive Java (JAR) qui contient l'image de distribution, vous installez l'environnement de service d'applications et vous êtes prêt à créer un serveur Liberty.

Pourquoi et quand exécuter cette tâche

Vous pouvez installer l'environnement de service d'applications profil Liberty en exécutant un fichier JAR, comme indiqué dans cette rubrique ou en utilisant le gestionnaire d'installation.

Lorsque vous exécutez le fichier JAR pour WebSphere Application Server pour installer profil Liberty, vous devez d'abord extraire le fichier JAR. Ensuite, extrayez le fichier JAR profil Liberty pour WebSphere eXtreme Scale. Si vous utilisez IBM Installation Manager pour installer WebSphere Application Server version 8.5 et obtenir le profil Liberty, vous devez utiliser le gestionnaire d'installation pour installer également WebSphere eXtreme Scale.

Cette tâche prend en charge les éditions suivantes :

- WebSphere Application Server Liberty Core
- WebSphere Application Server, édifions Base et Developer
- WebSphere Application Server, Network Deployment
- WebSphere Application Server for z/OS

Pour les informations de téléchargement des environnements de gestion d'applications et de mise en cache des données profil Liberty, voir la page des téléchargements de la communauté WASdev.

Procédure

1. Extrayez l'image de distribution du profil Liberty WebSphere Application Server vers le répertoire de votre choix.

   Cette image est fournie sous la forme d'un fichier JAR ; par exemple, `wlp-`*`édition`*`-8.6.0.0.jar`. Extrayez ce fichier JAR de l'une des manières suivantes :
   - Pour extraire l'image de distribution en utilisant l'assistant, exécutez `java -jar wlp-`*`édition`*`-8.6.0.0.jar`.
   - Pour extraire l'image de distribution en acceptant les conditions de la licence et les conditions d'installation silencieuse, exécutez `java -jar wlp-édition-8.6.0.0.jar -acceptLicense`.
   - Pour afficher toutes les options disponibles, exécutez `java -jar wlp-édition-8.6.0.0.jar -help`.

   Tous les fichiers de serveur d'applications sont stockés dans le répertoire `wlp`.
2. Facultatif : Définissez la variable `JAVA_HOME` de votre environnement.

   Le profil Liberty nécessite un JRE pour son exécution. Il ne partage pas le JDK ou le JRE que le profil complet WebSphere Application Server utilise. Vous pouvez définir l'emplacement du JDK ou du JRE en utilisant la propriété **JAVA_HOME** dans le fichier `server.env`, comme indiqué dans Personnalisation de l'environnement du profil Liberty. Sur les systèmes Linux et UNIX, vous pouvez définir la variable **JAVA_HOME** dans le fichier `.bashrc` de l'utilisateur ou ajouter le chemin du JDK ou du JRE à la variable d'environnement **PATH**. Sur un système Windows, vous pouvez aussi définir **JAVA_HOME** en tant que variable d'environnement du système ou ajouter le chemin du JDK ou du JRE à la variable **PATH** du système.

   Windows Sur un système Windows, par exemple, utilisez les commandes suivantes pour faire pointer la propriété **JAVA_HOME** sur l'emplacement où votre JDK est installé et pour ajouter le répertoire `/bin` de votre installation Java™ à la variable PATH :

   ```
   set JAVA_HOME=C:\Progra~1\Java\JDK16
   set PATH=%JAVA_HOME%\bin;%PATH%
   ```

   Remarques :
   - L'environnement d'exécution profil Liberty recherche la commande **java** dans l'ordre suivant : propriété **JAVA_HOME**, propriété **JRE_HOME** et propriété système **PATH**.
   - Pour plus d'informations sur les environnements pris en charge Java et savoir où vous les procurer, voir Niveaux Java minimaux pris en charge dans profil Liberty : Restrictions connues de l'environnement d'exécution.
3. Extrayez l'image de distribution WebSphere eXtreme Scale vers le répertoire dans lequel vous avez extrait le fichier `wlp-`*`édition`*`-8.6.0.0.jar`.

Cette image se trouve dans le fichier JAR `wxs-wlp_8.6.0.0.jar` en version 8.6 et `wxs-wlp_8.6.0.2.jar` en version 8.6 groupe de correctifs 2. Pour extraire l'image de distribution, exécutez le fichier JAR ; par exemple, exécutez la commande suivante, en fonction de votre version de eXtreme Scale:

```
java -jar wxs-wlp_8.6.0.0.jar -acceptLicense
```

```
java -jar wxs-wlp_8.6.0.2.jar -acceptLicense
```

Résultats

Si vous extrayez le fichier le fichier `wxs-wlp_8.6.0.0.jar` pour la version 8.6 et le fichier `wxs-wlp_8.6.0.2.jar` pour la version 8.6 groupe de correctifs 2, eXtreme Scale est installé par dessus le profil Liberty WebSphere Application Server lorsque vous extrayez les deux fichiers JAR vers le même répertoire.

Rubrique parent : Installation de profil Liberty

Identification et résolution des incidents liés à l'installation du produit

IBM® Installation Manager est un programme d'installation commun à de nombreux logiciels IBM qui vous permet d'installer cette version de WebSphere eXtreme Scale.

Résultats

Remarques sur la journalisation et le traçage :

- Pour consulter aisément les journaux, ouvrez le gestionnaire d'installation et cliquez sur **File > View Log**. Pour ouvrir un fichier journal individuel, il vous suffira alors de le sélectionner dans le tableau et de cliquer sur l'icône **Open log file**.
- Les journaux sont situés dans le répertoire `logs`, à l'emplacement des données applicatives d'Installation Manager. Exemple :
  - Windows **Installation par un administrateur :**

    ```
    C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
    ```

  - Windows **Installation par un non administrateur :**

    ```
    C:\Documents and Settings\nom_utilisateur\Application Data\IBM\Installation Manager
    ```

  - UNIX Linux **Installation par un administrateur :**

    ```
    /var/IBM/InstallationManager
    ```

  - UNIX Linux **Installation par un non administrateur :**

    ```
    rép_utilisateur/var/ibm/InstallationManager
    ```

- Les principaux fichiers journaux sont horodatés et conservés dans le répertoire `logs` au format XML. Ils peuvent donc être consultés à l'aide d'un navigateur Web standard.
- Le fichier `log.properties` du répertoire `logs` indique le niveau de journalisation ou de traçage appliqué par Installation Manager. Pour activer la fonction de traçage des plug-ins de WebSphere eXtreme Scale, par exemple, créez un fichier `log.properties` contenant les informations suivantes :

  ```
  com.ibm.ws=DEBUG
  com.ibm.cic.agent.core.Engine=DEBUG
  global=DEBUG
  ```

  Redémarrez Installation Manager au besoin. Installation Manager fournira ainsi les traces des plug-ins de WebSphere eXtreme Scale.

Remarques sur la résolution des problèmes :

- UNIX Linux Par défaut, certains systèmes HP-UX sont configurés pour ne pas utiliser de système DNS pour résoudre les noms d'hôte. Il est par conséquent possible qu'Installation Manager ne puisse pas se connecter à un référentiel externe.

  Vous pouvez vérifier la connexion au référentiel à l'aide de l'utilitaire Ping mais nslookup ne renverra aucun résultat.

  Demandez à votre administrateur système de configurer votre machine afin qu'elle utilise un système DNS ou utilisez l'adresse IP du référentiel.
- Dans certains cas, il peut être nécessaire d'ignorer les mécanismes de vérification existants dans Installation Manager.
  - Sur certains systèmes de fichiers réseau, l'espace disque peut ne pas être signalé correctement et il peut être nécessaire d'ignorer la vérification d'espace disque et de poursuivre l'installation.

    Pour désactiver la vérification d'espace disque, spécifiez la propriété système suivante dans le fichier `config.ini` du répertoire *`racine_install_IM`*`/eclipse/configuration` et redémarrez Installation Manager :

```
cic.override.disk.space=tailleunité
```

où *taille* est un chiffre entier et *unité* est laissé vide pour octets, a la valeur k pour kilo, m pour mégaoctets ou g pour gigaoctets. Par exemple :

```
cic.override.disk.space=120 (120 octets)
cic.override.disk.space=130k (130 kilooctets)
cic.override.disk.space=140m (140 mégaoctets)
cic.override.disk.space=150g (150 gigaoctets)
cic.override.disk.space=true
```

Installation Manager indique que la taille de l'espace disque est Long.MAX_VALUE. Au lieu d'afficher une grande quantité d'espace disque disponible, N/A s'affiche.

- Pour ignorer la vérification des prérequis pour le système d'exploitation, ajoutez `disableOSPrereqChecking=true` au fichier `config.ini` dans *racine_install_IM*`/eclipse/configuration` et redémarrez Installation Manager.

  Si vous devez utiliser l'une de ces méthodes, contactez le support IBM pour obtenir de l'aide et développer une solution n'impliquant pas d'ignorer les mécanismes de vérification d'Installation Manager.

- Pour plus d'informations sur l'utilisation d'Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

  Pour en savoir plus sur la dernière version d'Installation Manager, lisez les notes sur l'édition. Pour accéder aux notes sur l'édition, procédez comme suit :

  - Windows Cliquez sur **Démarrer > Programmes > IBM Installation Manager > Release Notes**.
  - UNIX Linux Accédez au sous-répertoire de la documentation dans le répertoire dans lequel Installation Manager est installé et ouvrez le fichier `readme.html`.

- Si une erreur fatale se produit lors de l'installation du produit, effectuez les étapes suivantes :
  - Faites une copie de sauvegarde de votre répertoire d'installation de produit actuel au cas où le service de support IBM aurait besoin de l'examiner ultérieurement.
  - Utilisez Installation Manager pour désinstaller tout ce que vous avez installé dans l'emplacement d'installation du produit (groupe de packages). Vous risquez de rencontrer des erreurs, mais vous pouvez les ignorer en toute sécurité.
  - Supprimez tout ce qui reste dans le répertoire d'installation du produit.
  - Utilisez Installation Manager pour réinstaller le produit dans le même emplacement ou dans un nouvel emplacement.

Remarque sur les informations de version et d'historique : Les commandes **`versionInfo`** et **`historyInfo`** retournent les informations de version et d'historique basées sur toutes les activités d'installation, de désinstallation, de mise à jour et de rétrogradation effectuées sur le système.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Mise à jour de WebSphere DataPower XC10 Appliance

Pour appliquer un pack de maintenance à votre environnement WebSphere DataPower XC10 Appliance, appliquez d'abord les éventuelles mises à jour du microprogramme. Vous pouvez ensuite appliquer des mises à jour à vos installations WebSphere eXtreme Scale Client.

1. Mise à jour du microprogramme
   Vous pouvez mettre à niveau le logiciel de votre IBM® WebSphere DataPower XC10 Appliance à l'aide des mises à jour du microprogramme. Téléchargez les nouvelles versions du microprogramme à partir du site Web Fix Central, puis mettez à jour le logiciel du dispositif.

2. Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server
   Lors de la migration de WebSphere Application Server vers une nouvelle version, vous pouvez également faire migrer la configuration de WebSphere eXtreme Scale Client vers la nouvelle installation WebSphere Application Server.

3. Installation des groupes de correctifs à l'aide d'IBM Installation Manager
   Vous pouvez utiliser IBM Installation Manager pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale Client. Les groupes de correctifs peuvent être installés à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

4. Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager
   Vous pouvez utiliser IBM Installation Manager pour rétrograder les offres de produit WebSphere eXtreme Scale Client vers une version précédente. Vous pouvez désinstaller les groupes de correctifs à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

5. 2.5+ Mise à niveau de WebSphere eXtreme Scale Client for .NET
   Pour mettre à niveau une installation existante de WebSphere eXtreme Scale Client for .NET, vous pouvez utiliser le programme d'installation. Ce dernier détecte l'installation existante et remplace les fichiers appropriés.

Mise à jour du microprogramme

Vous pouvez mettre à niveau le logiciel de votre IBM® WebSphere DataPower XC10 Appliance à l'aide des mises à jour du microprogramme. Téléchargez les nouvelles versions du microprogramme à partir du site Web Fix Central, puis mettez à jour le logiciel du dispositif.

Avant de commencer

Vous devez disposer des droits d'accès d'administrateur du dispositif.

Attendez que toutes les tâches actives soient terminées avant de lancer le processus de mise à jour du microprogramme. En effet, ce processus met fin à tous les travaux en cours d'exécution, ce qui pourrait entraîner alors une incohérence des données. Tenez-en compte avant de lancer le processus de mise à jour.

Important : Si vous installez le microprogramme pour la première fois sur un nouveau dispositif, vous devez exécuter la commande **`clear-all`** sur le dispositif une fois la mise à jour du microprogramme terminée. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Pourquoi et quand exécuter cette tâche

Vous pouvez mettre à jour le microprogramme du dispositif en téléchargeant une nouvelle mise à jour du microprogramme à partir du site IBM et en mettant à jour le dispositif avec le nouveau microprogramme. Vous pouvez exécuter la mise à jour du microprogramme dans l'interface utilisateur ou sur la ligne de commande.

ATTENTION :

Lors d'une mise à niveau vers la version 2.5 à partir de la version 2.0 ou d'une version antérieure, vous ne pouvez pas exécuter de mise à niveau de microprogramme sur une collectivité traitant une charge de travail. En effet, lors de la mise à niveau d'une collectivité, toutes les données chargées dans les grilles de données sont perdues. Quand vous exécutez des mises à niveau de microprogramme sur les dispositifs d'une collectivité, le dispositif mis à niveau ne redémarre pas en intégralité tant que tous les dispositifs de la collectivité ne sont pas mis à niveau. Les dispositifs restants qui ne sont pas encore mis à niveau prennent en charge les demandes. Vous devez terminer la mise à niveau du microprogramme d'un dispositif avant de démarrer le processus sur un autre dispositif de la collectivité. N'apportez pas de modifications de configuration tant que tous les dispositifs ne sont pas au même niveau de microprogramme.

Procédure

- **Téléchargez la mise à niveau du microprogramme :**

  Accédez au site IBM Fix Central et téléchargez une mise à jour du microprogramme dans votre système de fichiers local. Dans la page Fix Central, sélectionnez **WebSphere** comme **Groupe de produits** et **WebSphere DataPower XC10 Appliance** dans la liste des produits. Téléchargez la mise à jour de microprogramme appropriée pour votre type de matériel de dispositif :
  - Les mises à jour du microprogramme pour le type de dispositif 7199-92X correspondent à un fichier unique portant l'extension `.scrypt3`.

  Ces fichiers `scrypt` sont assortis d'une signature pour garantir l'intégrité de la mise à jour exécutée. Si vous utilisez l'interface utilisateur pour exécuter la mise à niveau du microprogramme, sauvegardez ce fichier sur l'ordinateur que vous utilisez pour accéder à l'interface utilisateur. Si vous utilisez l'interface de ligne de commande pour exécuter la mise à niveau du microprogramme, sauvegardez ce fichier sur un serveur auquel vous pouvez accéder à partir du dispositif.
- **Pour exécuter la mise à jour du microprogramme dans l'interface utilisateur :**
  1. Connectez-vous à l'interface utilisateur.
  2. Accédez au panneau Paramètres.
     - Dans l'interface utilisateur de WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres**.
     - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.
  3. Développez l'entrée **Microprogramme**. Le niveau du microprogramme installé sur le dispositif s'affiche.
  4. Installez une nouvelle mise à jour du microprogramme.
     a. Cliquez sur **Parcourir...** pour pouvoir sélectionner le fichier de mise à jour du microprogramme.
     b. Sélectionnez le fichier de mise à jour du microprogramme et cliquez sur **OK**.
     c. Cliquez sur **Mettre à niveau**. La durée du téléchargement de la mise à jour du

microprogramme dépend du débit de votre connexion. Un message s'affiche à l'issue du téléchargement pour vous aviser du démarrage de la mise à jour du microprogramme. Au début de cette mise à jour, le dispositif est redémarré sans que l'écran n'indique de progression ou n'affiche de modifications dans l'interface utilisateur. Les modifications ne sont pas affichées car votre session a pris fin au redémarrage du dispositif et l'interface utilisateur est indisponible au cours du processus de mise à niveau.

Important : Le dispositif redémarre plusieurs fois pendant le processus de mise à jour du microprogramme. N'interrompez pas le processus ou ne redémarrez pas manuellement le dispositif pendant la mise à jour du microprogramme.

Pour vérifier que la mise à jour du microprogramme s'est achevée, vous devez vous connecter à nouveau lorsque cette opération vous paraît terminée. La mise à jour du microprogramme dure en moyenne 10 à 15 minutes, mais elle peut prendre plus longtemps. Si vous souhaitez surveiller la progression de la mise à jour du microprogramme, utilisez le panneau d'état de démarrage de l'interface utilisateur du dispositif ou la commande **`start-progress`** de l'interface de ligne de commande. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.

- **Pour exécuter la mise à jour du microprogramme dans l'interface de ligne de commande :**

  1. Connectez-vous au dispositif à l'aide de la ligne de commande ou de la console série. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.
  2. Placez le fichier `.scrypt` sur le dispositif. A partir de l'interface de ligne de commande, exécutez la commande suivante :

     ```
     file get <url_fichier_scrypt> <fichier_microprogramme>
     ```

     Où *`url_fichier_scrypt`* représente au serveur local sur lequel vous avez sauvegardé le fichier `.scrypt` à partir de Fix Central, et *`fichier_microprogramme`* au nom du fichier `.scrypt` à utiliser sur le dispositif.
  3. Installez la mise à jour du microprogramme. Exécutez la commande suivante :

     ```
     firmware upgrade <fichier_microprogramme>
     ```

     Important : Pendant le processus de mise à niveau du microprogramme, le dispositif redémarre plusieurs fois. N'interrompez pas le processus ou ne redémarrez pas manuellement le dispositif pendant la mise à jour du microprogramme.
  4. Surveillez la progression du démarrage du dispositif.

     2.5+ Utilisez l'une des options suivantes :

     - 2.5+ Utilisez l'interface utilisateur du dispositif. Dans la barre d'adresse du navigateur Web, entrez l'URL et le port suivants : `https://<nom-hôte_dispositif>:9443/`. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.
     - Exécutez la commande **`start-progress`**. Lorsque cette commande renvoie `STARTED`, cela signifie que le dispositif est prêt à l'emploi.

Résultats

Le microprogramme du dispositif a été mis à jour. Pour vérifier si la mise à niveau du microprogramme a réussi, affichez le niveau actuel du microprogramme dans la section Microprogramme du panneau **Dispositif** > **Paramètres**. Ce panneau affiche également le type de modèle et le numéro de série du dispositif. Répétez ces étapes pour les autres dispositifs de la collectivité.

Que faire ensuite

Si vous installez le microprogramme pour la première fois sur un nouveau dispositif, vous devez exécuter la commande **`clear-all`** sur le dispositif une fois la mise à jour du microprogramme terminée. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique suivante : Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server

Tâches associées:
Démarrage rapide : Installation du matériel du dispositif
Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server

Lors de la migration de WebSphere Application Server vers une nouvelle version, vous pouvez également faire migrer la configuration de WebSphere eXtreme Scale Client vers la nouvelle installation WebSphere Application Server.

Avant de commencer

- On part du principe que WebSphere eXtreme Scale Version 7 et WebSphere eXtreme Scale Version 8 sont installés sur le même serveur.
- Faites migrer WebSphere Application Server Version 7 vers WebSphere Application Server Version 8. Pour plus d'informations, voir Migration des configurations de produit.
- Installez WebSphere eXtreme Scale Client version 8 dans votre installation WebSphere Application Server version 8. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server. Tous les scripts de migration de WebSphere eXtreme Scale Client doivent être exécutés à partir de WebSphere eXtreme Scale Client version 8.5 ou ultérieure. Par exemple, pour la migration de la version 7.x vers la version 8, exécutez les scripts de migration à partir du répertoire `<racine_install_WXS_v8>/bin`.

Pourquoi et quand exécuter cette tâche

Lorsque vous installez une nouvelle version de WebSphere Application Server dotée de l'intégration WebSphere eXtreme Scale, mettez d'abord à niveau WebSphere Application Server à l'aide du processus normal. Installez ensuite la nouvelle version de WebSphere eXtreme Scale Client sur votre nouvelle installation. Vous pouvez ensuite utiliser le script **xsmigration** pour déplacer les informations de configuration de WebSphere eXtreme Scale vers la nouvelle installation WebSphere Application Server.

Procédure

1. Faites migrer la configuration liée au gestionnaire de déploiement à partir de la version 7 vers la version 8.
    a. Exécutez le script de sauvegarde WebSphere Application Server. Pour plus d'informations, voir Commande WASPreUpgrade.
    b. Arrêtez le gestionnaire de déploiement.
    c. Accédez au serveur de gestionnaire de déploiement dans la configuration WebSphere eXtreme Scale Client, puis exécutez le script de migration.
        i. Accédez au répertoire `<racine_install_WXS_v8>/bin`
        ii. Exécutez la commande suivante :

```
xsmigration.bat|sh -targetwashome <WAS8x_HOME>
-sourcewashome <WAS7x_HOME> -targetprofilepath <WAS8x_DmgrProfile>
 -sourceprofilepath <WAS7x_DmgrProfile>
```

        où
        - *<WAS8x_HOME>* représente l'emplacement racine de l'installation WebSphere Application Server Version 8.x. Exemple : `/opt/IBM/WebSphere8`
        - *<WAS7x_HOME>* représente l'emplacement racine de l'installation WebSphere Application Server Version 7.x. Exemple : `/opt/IBM/WebSphere7`
        - *<WAS8x_DmgrProfile>* représente l'emplacement du profil de gestionnaire de déploiement WebSphere Application Server Version 8.x. Exemple : `/opt/IBM/WebSphere8/profiles/DMgr01`
        - *<WAS7x_DmgrProfile>* représente l'emplacement du profil de gestionnaire de déploiement WebSphere Application Server Version 7.x. Exemple : `/opt/IBM/WebSphere7/profiles/DMgr01`
2. Faites migrer la configuration liée au serveur d'applications à partir de la version 7 vers la version 8.
    a. Accédez au répertoire `<racine_install_WXS_v8>/bin`.
    b. Exécutez la commande suivante :

```
xsmigration.bat|sh -targetwashome <WAS8x_HOME>
-sourcewashome <WAS7x_HOME> -targetprofilepath <WAS8x_AppServerProfile>
 -sourceprofilepath <WAS7x_AppServerProfile>
```

    où
    - *<WAS8x_HOME>* représente l'emplacement racine de l'installation WebSphere Application

Server Version 8.x. Exemple : `/opt/IBM/WebSphere8`
- *<WAS7x_HOME>* représente l'emplacement racine de l'installation WebSphere Application Server Version 7.x. Exemple : `/opt/IBM/WebSphere7`
- *<WAS8x_AppServerProfile>* représente l'emplacement du profil de serveur d'applications WebSphere Application Server version 8.x. Exemple : `/opt/IBM/WebSphere8/profiles/AppServer01`
- *<WAS7x_AppServerProfile>* représente l'emplacement du profil de serveur d'applications WebSphere Application Server version 7.x. Exemple : `/opt/IBM/WebSphere7/profiles/AppServer01`

3. Redémarrez le gestionnaire de déploiement WebSphere Application Server version 8, puis synchronisez tous les noeuds gérés.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Mise à jour du microprogramme
Rubrique suivante : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Java

Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Vous pouvez utiliser IBM® Installation Manager pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale Client. Les groupes de correctifs peuvent être installés à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

Installation des groupes de correctifs à l'aide de l'interface graphique
Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure à l'aide de l'assistant d'IBM Installation Manager.

Installation des groupes de correctifs à l'aide de la ligne de commande
Vous pouvez utiliser IBM Installation Manager à partir de la ligne de commande pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale.

Installation des groupes de correctifs à l'aide d'un fichier de réponses
Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure avec IBM Installation Manager à l'aide d'un fichier de réponses.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server
Rubrique suivante : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Java

Installation des groupes de correctifs à l'aide de l'interface graphique

Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure à l'aide de l'assistant d'IBM® Installation Manager.

Avant de commencer

Mettez à niveau le microprogramme du dispositif avant de mettre à jour WebSphere eXtreme Scale Client. Pour plus d'informations, voir Mise à jour du microprogramme Prenez contact avec le centre de support logiciel IBM pour obtenir des informations sur les mises à niveau pour WebSphere eXtreme Scale autonome ou les offres de produit WebSphere eXtreme Scale for WebSphere Application Server. Les informations les plus récentes sont disponibles dans le centre de support logiciel IBM et dans Fix Central.

IBM Installation Manager sert à appliquer un package de maintenance de produit aux offres de produit suivantes :

- WebSphere eXtreme Scale Client dans un environnement autonome
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

Assurez-vous que l'emplacement du référentiel de service Web ou local est répertorié et vérifié ou que l'option **Search service repositories during installation and updates** est sélectionnée dans le panneau Repositories de la page des préférences Installation Manager. Pour plus d'informations sur l'utilisation des référentiels de service avec Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour mettre à niveau une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
2. Ouvrez une session sur votre système.
3. Arrêtez tous les serveurs et toutes les applications présents dans l'installation de WebSphere Application Server que vous voulez mettre à jour.
4. Démarrez Installation Manager.
5. Cliquez sur **Update**.

   Remarque : Si vous êtes invité à vous authentifier, utilisez l'ID IBM et le mot de passe vous permettant d'accéder aux sites Web protégés d'IBM.

6. Sélectionnez le groupe de packages à mettre à jour.

   Conseil : Si vous sélectionnez **Update all**, Installation Manager recherche dans tous les référentiels ajoutés et prédéfinis les mises à jour de tous les groupes de packages qu'il a installés. Utilisez cette fonction uniquement si vous contrôlez intégralement quels correctifs sont contenus dans les référentiels cible. Si vous créez et pointez vers un ensemble de référentiels personnalisés incluant uniquement les correctifs spécifiques que vous souhaitez installer, vous devez pouvoir utiliser cette fonction en toute confiance. Si vous activez la recherche dans les référentiels de service ou que vous installez les correctifs directement à partir d'autres référentiels Web en ligne, vous ne souhaitez peut-être pas sélectionner cette option afin de pouvoir sélectionner uniquement les correctifs à installer pour chaque offre dans des panneaux ultérieurs.

7. Cliquez sur **Next**.
8. Sélectionnez la version que vous voulez mettre à jour :
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
9. Sélectionnez les correctifs que vous voulez installer.

   Les correctifs recommandés sont sélectionnés par défaut.

   Vous pouvez choisir de n'afficher que les correctifs recommandés et de masquer les autres.

10. Cliquez sur **Next**.
11. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
12. Sélectionnez les fonctions facultatives que vous souhaitez intégrer à votre installation mise à jour.
13. Lisez le récapitulatif et cliquez sur **Update**.
    - Si l'installation a réussi, le programme affichera un message pour confirmer le succès de l'installation.
    - Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
14. Cliquez sur **Terminer**.
15. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Rubrique parent : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Désinstallation des groupes de correctifs à l'aide de l'interface graphique

Java

Installation des groupes de correctifs à l'aide de la ligne de commande

Vous pouvez utiliser IBM® Installation Manager à partir de la ligne de commande pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale.

Avant de commencer

Mettez à niveau le microprogramme du dispositif avant de mettre à jour WebSphere eXtreme Scale Client. Pour plus d'informations, voir Mise à jour du microprogramme. Prenez contact avec le centre de support logiciel IBM pour obtenir des informations sur les mises à niveau pour WebSphere eXtreme Scale autonome ou les offres de produit WebSphere eXtreme Scale for WebSphere Application Server. Les informations les plus récentes sont disponibles dans le centre de support logiciel IBM et dans Fix Central.

IBM Installation Manager sert à appliquer un package de maintenance de produit aux offres de produit suivantes :

- WebSphere eXtreme Scale Client dans un environnement autonome
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour mettre à niveau une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Pour obtenir la liste des groupes de correctifs et des correctifs temporaires disponibles pour WebSphere eXtreme Scale 8.5, ainsi que des informations spécifiques sur chaque correctif, procédez comme suit :
    a. Accédez à Fix Central.
    b. Sélectionnez **WebSphere** comme groupe de produits.
    c. Sélectionnez WebSphere eXtreme Scale comme produit.
    d. Sélectionnez **8.5** pour la version installée.
    e. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
    f. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
    g. Cliquez sur **Informations complémentaires** sous chaque correctif pour afficher des informations relatives au correctif.
    h. **Recommandation :** Notez le nom du groupe de correctifs que vous souhaitez installer.
2. Mettez à jour WebSphere eXtreme Scale version 8.5 à l'aide du groupe de correctifs en utilisant la procédure ci-dessous.
    - Téléchargez sur Fix Central le fichier contenant le groupe de correctifs, puis utilisez la mise à jour locale.

      Vous pouvez télécharger à partir de Fix Central un fichier compressé contenant le groupe de correctifs. Chaque fichier de groupe de correctifs compressé contient un référentiel Installation Manager pour le groupe de correctifs et comporte généralement une extension .zip. Après avoir téléchargé et extrait le fichier de groupe de correctifs, utilisez Installation Manager pour mettre à jour WebSphere Application Server version 8.x à l'aide du groupe de correctifs.

      a. Pour télécharger le groupe de correctifs, procédez comme suit :
          i. Accédez à Fix Central.
          ii. Sélectionnez **WebSphere** comme groupe de produits.
          iii. Sélectionnez **WebSphere eXtreme Scale** comme produit.
          iv. Sélectionnez **8.5** pour la version installée.
          v. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
          vi. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
          vii. Sélectionnez le groupe de correctifs à télécharger, puis cliquez sur **Continuer**.
          viii. Sélectionnez vos options de téléchargement, puis cliquez sur **Continuer**.

ix. Cliquez sur **J'accepte** pour accepter les dispositions.
x. Cliquez sur **Télécharger maintenant** pour télécharger le groupe de correctifs.
xi. Transférez le fichier compressé au format binaire vers le système sur lequel il sera installé.
xii. Extrayez les fichiers de référentiel compressés dans un répertoire de votre système.

b. Pour installer un groupe de correctifs à partir d'un fichier téléchargé, procédez comme suit :

i. Ouvrez une session sur votre système.

ii. Arrêtez tous les processus en cours d'exécution dans votre environnement. Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.

iii. Accédez au répertoire *binaires_Installation_Manager*`/eclipse/tools`, où *binaires_Installation_Manager* représente le répertoire racine d'installation d'Installation Manager.

iv. Installez le groupe de correctifs.

UNIX Linux

```
./imcl install ID_offre_version_offre,ID_fonction_facultative
  -installationDirectory emplacement_installation_produit
  -repositories emplacement_fichiers_étendus
  -acceptLicense
```

Windows

```
imcl.exe install ID_offre_version_offre,ID_fonction_facultative
  -installationDirectory emplacement_installation_produit
  -repositories emplacement_fichiers_étendus
  -acceptLicense
```

Conseils :

- La *version_offre*, qui peut, le cas échéant, être associée à l'ID offre à l'aide d'un trait de soulignement, est une version spécifique de l'offre à installer (`8.5.0.20110503_0200`, par exemple).
  - Si la *version_offre* n'est **pas** spécifiée, la version la plus récente de l'offre et **tous** les correctifs temporaires de cette version sont installés.
  - Si la *version_offre* est spécifiée, la version indiquée de l'offre est installée et **aucun** correctif temporaire de cette version n'est installé.

  La version de l'offre peut être rattachée à la fin de l'ID offre à l'aide d'un trait de soulignement lorsque vous exécutez la commande suivante au niveau du référentiel :

  ```
  imcl listAvailablePackages -repositories référentiel_source
  ```

- Vous pouvez également spécifier `none`, `recommended` ou `all` avec l'argument `-installFixes` afin d'indiquer les correctifs temporaires que vous souhaitez installer avec l'offre.
  - Si la version de l'offre n'est **pas** spécifiée, l'option `-installFixes` est paramétrée par défaut sur `all`.
  - Si la version de l'offre est spécifiée, l'option `-installFixes` est paramétrée par défaut sur `none`.
- Vous pouvez ajouter une liste de fonctions séparées par des virgules. Si aucune liste de fonctions n'est spécifiée, les fonctions par défaut sont installées.

v. **Facultatif :** Répertoriez tous les packages installés pour vérifier l'installation :

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Rubrique parent : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Désinstallation des groupes de correctifs à l'aide de la ligne de commande

Java

Installation des groupes de correctifs à l'aide d'un fichier de réponses

Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure avec IBM® Installation Manager à l'aide d'un fichier de réponses.

Avant de commencer

Conseil : Comme alternative à la procédure décrite dans cet article, Installation Manager vous permet d'utiliser la commande **`updateAll`** dans un fichier de réponses ou sur la ligne de commande pour rechercher et mettre à jour tous les packages installés. Utilisez cette commande uniquement si vous contrôlez intégralement quels correctifs sont contenus dans les référentiels cible. Si vous créez et pointez vers un ensemble de référentiels personnalisés incluant uniquement les correctifs spécifiques que vous souhaitez installer, vous devez pouvoir utiliser cette commande en toute confiance. Si vous activez la recherche dans les référentiels de service ou que vous installez les correctifs directement à partir d'autres référentiels Web en ligne, vous ne souhaitez peut-être pas sélectionner cette option afin de pouvoir sélectionner uniquement les correctifs à installer à l'aide de l'option -installFixes avec la commande **`install`** sur la ligne de commande ou à l'aide de l'attribut installFixes dans un fichier de réponses.

Procédure

1. Pour obtenir la liste des groupes de correctifs et des correctifs temporaires disponibles pour WebSphere eXtreme Scale Client et des informations sur chaque correctif, procédez comme suit.
    a. Accédez à Fix Central.
    b. Sélectionnez **WebSphere** comme groupe de produits.
    c. Sélectionnez WebSphere eXtreme Scale Client comme produit.
    d. Sélectionnez **8.x** pour la version installée.
    e. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
    f. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
    g. Cliquez sur **Informations complémentaires** sous chaque correctif pour afficher des informations relatives au correctif.
    h. **Recommandation :** Notez le nom du groupe de correctifs que vous souhaitez installer.
2. Mettez à jour WebSphere eXtreme Scale Client avec le correctif en procédant comme suit.
    - Téléchargez sur Fix Central le fichier contenant le groupe de correctifs, puis utilisez la mise à jour locale.

      Vous pouvez télécharger à partir de Fix Central un fichier compressé contenant le groupe de correctifs. Chaque fichier de groupe de correctifs compressé contient un référentiel Installation Manager pour le groupe de correctifs et comporte généralement une extension .zip. Après avoir téléchargé et extrait le fichier de groupe de correctifs, utilisez le gestionnaire d'installation pour mettre à jour WebSphere eXtreme Scale Client avec le groupe de correctifs.

      a. Pour télécharger le groupe de correctifs, procédez comme suit :
          i. Accédez à Fix Central.
          ii. Sélectionnez **WebSphere** comme groupe de produits.
          iii. Sélectionnez **WebSphere eXtreme Scale Client** comme produit.
          iv. Sélectionnez **8.6** comme version installée.
          v. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
          vi. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
          vii. Sélectionnez le groupe de correctifs à télécharger, puis cliquez sur **Continuer**.
          viii. Sélectionnez vos options de téléchargement, puis cliquez sur **Continuer**.
          ix. Cliquez sur **J'accepte** pour accepter les dispositions.

x. Cliquez sur **Télécharger maintenant** pour télécharger le groupe de correctifs.

xi. Transférez le fichier compressé au format binaire vers le système sur lequel il sera installé.

xii. Extrayez les fichiers de référentiel compressés dans un répertoire de votre système.

b. Effectuez les actions suivantes :

i. Ouvrez une session sur votre système.

ii. Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

iii. Accédez au répertoire *`binaires_Installation_Manager`*`/eclipse/tools`, où *`binaires_Installation_Manager`* représente le répertoire racine d'installation d'Installation Manager.

iv. Installez le groupe de correctifs à l'aide d'un fichier de réponses.

Par exemple :

- Windows **Administrateur ou non administrateur :**

```
imcl.exe -acceptLicense
  input C:\temp\update_response_file.xml
  -log C:\temp\update_log.xml
  -keyring C:\IM\im.keyring
```

- UNIX Linux **Administrateur :**

```
./imcl -acceptLicense
  input /var/temp/update_response_file.xml
  -log /var/temp/update_log.xml
  -keyring /var/IM/im.keyring
```

- UNIX Linux **Non administrateur :**

```
./imcl -acceptLicense
  input rép_base_utilisateur/var/temp/update_response_file.xml
  -log rép_base_utilisateur/var/temp/update_log.xml
  -keyring rép_base_utilisateur/var/IM/im.keyring
```

Rubrique parent : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Désinstallation des groupes de correctifs à l'aide de fichiers de réponses

Java

Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Vous pouvez utiliser IBM® Installation Manager pour rétrograder les offres de produit WebSphere eXtreme Scale Client vers une version précédente. Vous pouvez désinstaller les groupes de correctifs à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

Désinstallation des groupes de correctifs à l'aide de l'interface graphique
Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de l'interface graphique d'IBM Installation Manager.

Désinstallation des groupes de correctifs à l'aide de la ligne de commande
Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de la ligne de commande d'IBM Installation Manager.

Désinstallation des groupes de correctifs à l'aide de fichiers de réponses
Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente avec IBM Installation Manager à l'aide d'un fichier de réponses.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Installation des groupes de correctifs à l'aide d'IBM Installation Manager
Rubrique suivante : 2.5+ Mise à niveau de WebSphere eXtreme Scale Client for .NET

Java

Désinstallation des groupes de correctifs à l'aide de l'interface graphique

Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de l'interface graphique d'IBM® Installation Manager.

Avant de commencer

Pendant le processus de rétrogradation, Installation Manager doit accéder aux fichiers de la version précédente du package. Par défaut, ces fichiers sont enregistrés sur votre ordinateur lors de l'installation d'un package. Si vous modifiez le paramétrage par défaut ou que supprimez les fichiers enregistrés, Installation Manager a besoin d'accéder au référentiel utilisé pour l'installation de la version précédente.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour rétrograder une installation et ajouter ou supprimer une fonction.

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
    - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
2. Démarrez Installation Manager.
3. Cliquez sur **Rétrograder**.
4. Sélectionnez le groupe de packages à rétrograder.
5. Cliquez sur **Suivant**.
6. Sélectionnez la version que vous voulez rétrograder.
7. Cliquez sur **Suivant**.
8. Lisez le récapitulatif et cliquez sur **Rétrograder**.
    - Si la rétrogradation aboutit, le programme affiche un message pour confirmer le succès de la rétrogradation.
    - Dans le cas contraire, cliquez sur **View Log File** pour corriger l'erreur.
9. Cliquez sur **Terminer**.
10. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Rubrique parent : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Installation des groupes de correctifs à l'aide de l'interface graphique

Java

Désinstallation des groupes de correctifs à l'aide de la ligne de commande

Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de la ligne de commande d'IBM® Installation Manager.

Avant de commencer

Restriction : Pour utiliser cette procédure, Installation Manager version 1.5 ou ultérieure doit être installé sur votre système.

Pendant le processus de rétrogradation, Installation Manager doit accéder aux fichiers de la version précédente du package. Par défaut, ces fichiers sont enregistrés sur votre ordinateur lors de l'installation d'un package. Si vous modifiez le paramétrage par défaut ou que supprimez les fichiers enregistrés, Installation Manager a besoin d'accéder au référentiel utilisé pour l'installation de la version précédente.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour rétrograder une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Facultatif : Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

   Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

   Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

2. Ouvrez une session sur votre système.
3. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
4. Accédez au sous-répertoire `eclipse/tools` dans le répertoire où vous avez installé Installation Manager.
5. Utilisez la commande **`imcl`** pour rétrograder le produit.

   UNIX Linux

   ```
   ./imcl rollback ID_offre_version_offre
     -repositories référentiel_source
     -installationDirectory répertoire_installation
     -preferences clé_préférence=valeur
     -properties clé_propriété=valeur
     -keyring fichier_de_clés -password mot_de_passe
     -acceptLicense
   ```

   Windows

   ```
   imcl.exe rollback ID_offre_version_offre
   -repositories référentiel_source
   -installationDirectory répertoire_installation
   -preferences clé_préférence=valeur
   -properties clé_propriété=valeur
   -keyring fichier_de_clés -password mot_de_passe
   -acceptLicense
   ```

   Conseils :

   - La *version_offre*, qui peut, le cas échéant, être associée à l'ID offre à l'aide d'un trait de soulignement, est une version spécifique de l'offre à rétrograder (`8.5.0.20110503_0200`, par exemple).
     - Si la *version_offre* n'est **pas** spécifiée, l'installation est rétrogradée à la version précédemment installée de l'offre et **tous** les correctifs temporaires de cette version sont

installés.

- Si la *version_offre* est spécifiée, l'installation est rétrogradée à la version précédente indiquée de l'offre et **aucun** correctif temporaire de cette version n'est installé.

La version de l'offre peut être rattachée à la fin de l'ID offre à l'aide d'un trait de soulignement dans la section Package du rapport qui est généré lors de l'exécution de la commande **historyInfo** ou **genHistoryReport** à partir du répertoire *racine_serveur_app*/bin.

Pour plus d'informations sur l'utilisation d'Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

6. Facultatif : Répertoriez tous les packages installés pour vérifier la rétrogradation.

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Rubrique parent : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Installation des groupes de correctifs à l'aide de la ligne de commande

Java

Désinstallation des groupes de correctifs à l'aide de fichiers de réponses

Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente avec IBM® Installation Manager à l'aide d'un fichier de réponses.

Avant de commencer

Pendant le processus de rétrogradation, Installation Manager doit accéder aux fichiers de la version précédente du package. Par défaut, ces fichiers sont enregistrés sur votre ordinateur lors de l'installation d'un package. Si vous modifiez le paramétrage par défaut ou que supprimez les fichiers enregistrés, Installation Manager a besoin d'accéder au référentiel utilisé pour l'installation de la version précédente.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour rétrograder une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Facultatif : Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

   Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

   Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

2. Ouvrez une session sur votre système.
3. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
4. Utilisez un fichier de réponses pour rétrograder le produit.

   Accédez au sous-répertoire `eclipse/tools` du répertoire dans lequel vous avez installé Installation Manager, puis rétrogradez le produit.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     imcl.exe
       input C:\temp\rollback_response_file.xml
       -log C:\temp\rollback_log.xml
       -keyring C:\IM\im.keyring
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./imcl
       input /var/temp/rollback_response_file.xml
       -log /var/temp/rollback_log.xml
       -keyring /var/IM/im.keyring
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./imcl
       input rép_base_utilisateur/var/temp/rollback_response_file.xml
       -log rép_base_utilisateur/var/temp/rollback_log.xml
       -keyring rép_base_utilisateur/var/IM/im.keyring
     ```

   Remarque : Le programme peut générer des instructions post-installation importantes dans la sortie standard.

   Pour plus d'informations sur l'utilisation d'Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

5. Facultatif : Répertoriez tous les packages installés pour vérifier la rétrogradation.

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Rubrique parent : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Installation des groupes de correctifs à l'aide d'un fichier de réponses

.NET

Mise à niveau de WebSphere eXtreme Scale Client for .NET

Pour mettre à niveau une installation existante de WebSphere eXtreme Scale Client for .NET, vous pouvez utiliser le programme d'installation. Ce dernier détecte l'installation existante et remplace les fichiers appropriés.

Avant de commencer

- Mettez d'abord à jour le microprogramme de votre dispositif avant de mettre à niveau WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Mise à jour du microprogramme.
- Téléchargez la mise à niveau de WebSphere eXtreme Scale Client for .NET. Les informations les plus récentes sont disponibles dans le centre de support logiciel IBM et dans Fix Central. Les téléchargements disponibles concernent aussi bien les nouvelles installations que les mises à niveau.

Pourquoi et quand exécuter cette tâche

La procédure d'installation remplace immédiatement votre installation existante.

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
2. Utilisez l'assistant pour installer la mise à niveau de WebSphere eXtreme Scale Client for .NET. Lorsque vous utilisez l'assistant d'installation et qu'il détecte une installation précédente, vous devez confirmer que vous voulez la mettre à niveau. Le panneau de progression de l'assistant indique les références de la version précédente et de la version de mise à niveau.

Résultats

L'ensemble du code WebSphere eXtreme Scale Client for .NET existant est remplacé sur le disque et dans le cache GAC (Global Assembly Cache). Le fichier de règles publié est installé sur le disque et dans le cache GAC, en remplacement du fichier existant.

Que faire ensuite

- Configurez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de WebSphere eXtreme Scale Client for .NET.
- Développez vos applications .NET. Pour plus d'informations, voir Développement d'applications de grille de données avec les API .NET.

.NET **Création d'une installation côte à côte de groupes de correctifs pour WebSphere eXtreme Scale Client for .NET**
Lorsque vous créez une installation côte à côte, vous pouvez utiliser plusieurs versions des groupes de correctifs pour WebSphere eXtreme Scale Client for .NET sur le même serveur. Ainsi, vos applications .NET existantes qui ont été générées avec la version précédente peuvent continuer à s'exécuter avec la version précédente du client.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

.NET

Création d'une installation côte à côte de groupes de correctifs pour WebSphere eXtreme Scale Client for .NET

Lorsque vous créez une installation côte à côte, vous pouvez utiliser plusieurs versions des groupes de correctifs pour WebSphere eXtreme Scale Client for .NET sur le même serveur. Ainsi, vos applications .NET existantes qui ont été générées avec la version précédente peuvent continuer à s'exécuter avec la version précédente du client.

Avant de commencer

- Vous devez disposer d'un système équipé d'une version précédente de WebSphere eXtreme Scale Client for .NET, et d'un système distinct sur lequel vous installerez le groupe de correctifs plus récent.
- Installez la version plus récente de WebSphere eXtreme Scale Client for .NET sur un système distinct de vos systèmes de production. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client for .NET.

Pourquoi et quand exécuter cette tâche

Lorsque vous installez un groupe de correctifs de WebSphere eXtreme Scale Client for .NET, l'installation existante est remplacée par la nouvelle version. En fonction des caractéristiques de votre environnement, il se peut que vous préfériez effectuer des tests avant de procéder à la mise à niveau vers cette nouvelle version. Il se peut aussi que vous souhaitiez que certaines de vos applications utilisent la version précédente, et d'autres la version la plus récente. Installer manuellement WebSphere eXtreme Scale Client for .NET vous permet d'utiliser plusieurs versions en parallèle.

Lorsque vous installez une nouvelle édition importante de WebSphere eXtreme Scale Client for .NET, cette installation côte à côte a lieu automatiquement.

Procédure

1. A partir du système équipé de l'installation la plus récente, copiez l'ensemble du répertoire d'installation *net_client_home*, y compris tous ses sous-répertoires. Placez ces fichiers dans un répertoire distinct de celui contenant l'installation existante sur le système cible équipé de la version précédente. Ce répertoire est désigné sous le nom *base_côte-à-côte*.
2. Installez manuellement les assemblages WebSphere eXtreme Scale Client for .NET les plus récents dans le GAC (global assembly cache) à partir du répertoire *base_côte-à-côte*/bin. Après cette installation, le GAC contient l'ancienne et la nouvelle version des assemblages WebSphere eXtreme Scale Client for .NET. La procédure d'installation manuelle des assemblages dans le GAC dépend de la version de Windows et de l'infrastructure .NET installée sur votre système. Par exemple, vous pouvez utiliser l'explorateur Windows pour copier les fichiers des assemblages dans le répertoire %systemroot%\assembly. Vous pouvez également utiliser gacutil.exe, un utilitaire téléchargeable à partir du site Web de Microsoft.
3. Mettez à jour vos applications WebSphere eXtreme Scale Client for .NET pour qu'elles utilisent la nouvelle version. Utilisez l'une des options suivantes :
   - Ajoutez un élément de redirection d'assemblage au fichier de configuration de chaque application à mettre à jour. Cet élément redirige toute référence à l'ancien assemblage WebSphere eXtreme Scale Client for .NET vers le nouvel assemblage.

     ```
     <runtime>
       <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
          <dependentAssembly>
             <assemblyIdentity name="IBM.WebSphere.Caching"
     publicKeyToken="b439a24ee43b0816" />
             <bindingRedirect oldVersion="8.6.0.0-8.6.0.1" newVersion="8.6.0.2" />

          </dependentAssembly>
       </assemblyBinding>
     </runtime>
     ```

   - Régénérez votre application WebSphere eXtreme Scale Client for .NET en incluant le nouveau fichier d'assemblage dans la liste des références d'assemblage pour votre projet d'application.

Résultats

- Les applications ainsi mises à jour utiliseront la nouvelle version de WebSphere eXtreme Scale Client for .NET, tandis que les autres continueront d'utiliser la version précédente.
- Les fichiers journaux des applications qui utilisent la nouvelle version se trouvent dans le même

répertoire que les fichiers journaux des applications qui utilisent l'ancienne version.
- La configuration du client par défaut est toujours obtenue à partir de l'ancienne installation de WebSphere eXtreme Scale Client for .NET (répertoire *net_client_home*/config). Pour utiliser un autre fichier de propriétés, transmettez explicitement son chemin d'accès à l'API Connect().

Avertissement : Vous devez manuellement désinstaller vos modifications. Les installations manuelles ne sont pas mises à niveau par le programme d'installation de WebSphere eXtreme Scale Client for .NET.

Rubrique parent : 2.5+ Mise à niveau de WebSphere eXtreme Scale Client for .NET

Configuration de votre dispositif

Après avoir initialisé le dispositif à l'aide de la console série, vous devez le configurer à l'aide de l'interface utilisateur pour activer ses fonctionnalités.

Avant de commencer

- Installez et initialisez le matériel du dispositif. Pour plus d'informations, voir Installation de WebSphere DataPower XC10 Appliance.
- Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Le processus d'initialisation à l'aide de la console série prépare le dispositif pour son administration à partir de l'interface utilisateur. Pour préparer le dispositif pour son utilisation, vous devez configurer des paramètres supplémentaires.

Remarque : Tous ces paramètres sont importants, mais l'ajout correct d'un serveur DNS (Domain Name System), d'un serveur NTP (Network Time Protocol) et la configuration de la distribution du courrier revêt une importance toute particulière. Ces tâches doivent être effectuées avant toute autre utilisation.

Sécurité
Vous pouvez configurer plusieurs éléments de sécurité dans le dispositif, notamment la sécurité de l'interface utilisateur et du transport.

Gestion des utilisateurs et des groupes
Des utilisateurs et des groupes sont fournis pour gérer les accès à votre dispositif WebSphere DataPower XC10 Appliance par chaque individu. Vous pouvez utiliser des groupes d'utilisateurs pour appliquer des autorisations à des groupes d'utilisateurs.

Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Lors de la connexion série initiale, vous avez configuré l'interface Ethernet *mgmt* afin de connecter le dispositif à votre réseau. Vous pouvez définir des ports Ethernet privés supplémentaires dans l'interface utilisateur.

Importation et exportation de configurations
Lorsque vous configurez un nouveau dispositif, vous pouvez exporter et stocker les paramètres de configuration pour ce dispositif. Si ultérieurement des modifications sont effectuées qui nécessitent la suppression ou la réinstallation du dispositif, vous pouvez alors importer les données de configuration stockées sans perdre les paramètres de configuration.

Gestion du serveur DNS (Domain Name System)
Un serveur DNS (Domain Name System) est requis pour le système IBM® WebSphere DataPower XC10 Appliance étant donné que les services de recherche DNS sont utilisés pour la communication. Vous devez spécifier ce serveur DNS lors de l'initialisation du dispositif.

Mappage d'adresses IP vers des noms d'hôte
Avant que les informations relatives à une adresse puissent être utilisées pour créer une connexion dans un protocole de réseau TCP/IP, l'adresse IP doit être associée à un nom d'hôte. Vous pouvez résoudre une adresse IP en nom d'hôte en éditant le fichier etc/hosts du dispositif. Vous pouvez éditer le fichier etc/hosts à partir de l'interface utilisateur.

Gestion des paramètres de date et heure
Utilisez des serveurs NTP (Network Time Protocol) pour maintenir la synchronisation de la date et de l'heure entre les différents dispositifs de votre collectivité.

Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
La fonction de distribution du courrier du dispositif est utilisée afin de redéfinir les mots de passe des utilisateurs. Lorsqu'un utiliseur demande un nouveau mot de passe, il le reçoit dans un courrier électronique généré par le dispositif.

Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Le système WebSphere DataPower XC10 Appliance peut être redémarré ou arrêté à partir de l'interface utilisateur.

Sécurité

Vous pouvez configurer plusieurs éléments de sécurité dans le dispositif, notamment la sécurité de l'interface utilisateur et du transport.

IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
IBM WebSphere DataPower XC10 Appliance permet de contrôler l'accès au dispositif et aux données grille de données contenues dans le dispositif.

Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Une grande partie des fonctions de sécurité offertes par WebSphere DataPower XC10 Appliance sont déjà intégrées dans le dispositif lors de sa fabrication. D'autres paramètres sont inclus pour proposer des options de sécurité supplémentaires pour votre environnement.

Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Si vous le souhaitez, vous pouvez utiliser un annuaire LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs sur votre système IBM WebSphere DataPower XC10 Appliance.

Gestion des utilisateurs et des groupes
Des utilisateurs et des groupes sont fournis pour gérer les accès à votre dispositif WebSphere DataPower XC10 Appliance par chaque individu. Vous pouvez utiliser des groupes d'utilisateurs pour appliquer des autorisations à des groupes d'utilisateurs.

Activation de la sécurité pour les grilles de données
Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

Java **Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.**
Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Java **Configuration d'une application de grille de données pour l'utilisation de l'authentification client**
Si la grille de données que vous configurez pour l'application utilise la sécurité, vous devez configurer un fichier `client.properties` comportant des paramètres à transmettre à l'application de grille de données.

Configuration de TLS pour les applications de grille de données
Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Passerelle REST : Configuration de la sécurité
Pour accéder à une grille de données via la passerelle REST, l'utilisateur doit être authentifié sur WebSphere DataPower XC10 Appliance, que la sécurité de la grille de données ait été activée ou non. Le client d'application doit toujours fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP. Pour accéder à des grilles de données via la passerelle REST, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

IBM® WebSphere DataPower XC10 Appliance permet de contrôler l'accès au dispositif et aux données grille de données contenues dans le dispositif.

Sécurité du dispositif

Plusieurs caractéristiques critiques contribuent à sécuriser le dispositif, notamment :

Le dispositif est implanté dans un boîtier résistant aux intrusions

Un commutateur de détection d'intrusion présent sur le boîtier est sous surveillance permanente. Si le commutateur est déclenché, le dispositif ne démarre plus. Le dispositif doit alors être envoyé à IBM pour permettre son redémarrage. Des éléments supplémentaires, comme des vis anti-intrusion, sont également inclus pour décourager l'ouverture du boîtier. La conception du dispositif permet d'accéder aux éléments remplaçables par l'utilisateur depuis l'arrière du dispositif sans ouvrir le boîtier.

Aucun accès au système d'exploitation n'est possible via un interpréteur de commandes.

Le système d'exploitation du dispositif ne comporte pas d'interpréteur de commandes. L'absence d'interpréteur de commandes a été conçue délibérément pour réduire le périmètre de vulnérabilité du dispositif. Un seul ID utilisateur du système d'exploitation est utilisé sur le dispositif. Etant donné l'absence d'interpréteur de commandes, il n'est pas possible de se connecter depuis l'extérieur du dispositif avec un ID utilisateur.

Pas de possibilité d'exécution de logique soumise par l'utilisateur sur le dispositif

Le dispositif ne fournit pas la possibilité à un utilisateur de télécharger un script ou un code exécutable. L'unique exception à cette règle concerne une mise à jour du microprogramme du système au cours de laquelle vous exécutez un script pour installer le microprogramme mis à jour dans le dispositif. Ces mises à jour comportent, par précaution, une signature de l'éditeur du microprogramme. Aucun logiciel non accrédité soumis par l'utilisateur ne peut être exécuté sur le dispositif.

Grille de données : Sécurité

Vous pouvez gérer l'accès aux informations contenues dans vos grilles de données. Si vous n'activez pas la sécurité sur votre grille de données, les informations des grille de données sont accessibles par toutes les applications. Vous pouvez activer la sécurité générale sur une grille de données, afin d'autoriser tous les utilisateurs possédant un compte et un mot de passe sur le dispositif à accéder à la grille de données. Vous pouvez également restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs en activant des autorisations pour la grille de données.

TLS (Transport Layer Security)

Vous pouvez utiliser TLS pour protéger les grilles de données et l'interface utilisateur en configurant un fichier de clés, un fichier de clés certifiées et des alias de certificat. Les paramètres TLS s'appliquent à tous les dispositifs de la collectivité.

Utilisateurs et groupes d'utilisateurs

Vous pouvez définir des permissions pour les utilisateurs et les groupes d'utilisateurs. Ces permissions peuvent s'appliquer à l'administration du dispositif et à la sécurité de la grille de données.

Rubrique parent : Sécurité

Concepts associés:
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration de la sécurité du client

Référence associée:
Passerelle REST : Configuration de la sécurité
Fichier de propriétés du client

Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Avant de commencer

- Vous pouvez configurer TLS avec la version 1.0.0.4 ou une version suivante.
- Vous devez utiliser le correctif WebSphere eXtreme Scale Client version 7.1 ou suivante.
- Vous devez disposer de droits d'accès d'administrateur du dispositif.
- Vous devez disposer d'un fichier de clés ou d'un fichier de clés certifiées avec les mots de passe associés à ajouter à la configuration du dispositif.
- Pour modifier le fichier de clés certifiées existant, vous pouvez le télécharger à partir du dispositif.
- Vous devez mettre à jour le fichier de clés certifiées à l'aide des certificats publics des clients. Le dispositif doit accréditer les clients qui se connectent.
- Le fichier de clés certifiées fourni doit inclure un certificat public correspondant à une entrée dans le fichier de clés. Les alias de certificat issus du fichier de clés doivent être accrédités dans le fichier de clés certifiées à fournir sous la forme d'une option de configuration possible pour l'alias de certificat du dispositif.
- Le paramètre de sécurité globale WebSphere Application Server détermine comment le serveur tente de se connecter à WebSphere DataPower XC10 Appliance :
  - Lorsque le paramètre de sécurité globale est désactivé, des tentatives de connexion sur TCP/IP sont effectuées.
  - Lorsque le paramètre de sécurité globale est activé, vous devez ajouter le certificat public du dispositif aux fichiers de clés certifiées de WebSphere Application Server.

  Si le paramètre **TLS requis** de votre WebSphere DataPower XC10 Appliance est configuré, vous devez activer la sécurité globale. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Les paramètres TLS s'appliquent à l'interface utilisateur et aux grilles de données. Les paramètres sont appliqués à tous les dispositifs de la collectivité.

Procédure

1. **Requise pour WebSphere Application Server :** Ajoutez le certificat public du dispositif aux fichiers de clés certifiées par défaut de WebSphere Application Server.
   - **Si vous utilisez le fichier de clés certifiées par défaut du dispositif :**

     Exécutez le script **addXC10PublicCert.py** à partir du répertoire *racine_was*/bin du gestionnaire de déploiement. Utilisez la syntaxe suivante :

     ```
     wsadmin -lang jython -f addXC10PublicCert.py
     ```

   - **Si vous utilisez des clés personnalisées pour le dispositif :**

     Exécutez le script **addXC10PublicCert.py** à partir du répertoire *racine_was*/bin du gestionnaire de déploiement, à l'aide de l'option de ligne de commande **-certPath**. La valeur de l'option de ligne de commande **-certPath** est l'emplacement disque du certificat public qui correspond à l'alias configuré pour le fichier de clés certifiées dans le dispositif.

     ```
     wsadmin -lang jython -f addXC10PublicCert.py -certPath ./trustStore.jks
     ```

2. **Requise pour WebSphere Application Server :** Téléchargez le fichier de clés certifiées du dispositif et les certificats publics de WebSphere Application Server, puis exécutez l'utilitaire keytool pour ajouter le certificat dans le fichier de clés certifiées. Cet outil met à jour le fichier de clés certifiées du dispositif pour inclure les certificats émanant de WebSphere Application Server.
   a. Si vous utilisez le fichier de clés certifiées par défaut du dispositif, téléchargez le fichier de clés certifiées actif. Cliquez sur **Collectivité** > **Paramètres** > **Transport Layer Security (TLS)**. Cliquez sur **Télécharger un fichier de clés certifiées actif** et rappelez-vous l'emplacement dans lequel vous avez enregistré le fichier sur disque, par exemple dans le répertoire `/downloads/trustStore.jks`.
   b. Extrayez le certificat public de WebSphere Application Server.
      i. Dans la console d'administration de WebSphere Application Server, cliquez sur **Sécurité** > **Certificat SSL et gestion des clés** > **Fichiers de clés et certificats**.

ii. Dans **Utilisations des fichiers de clés**, sélectionnez **Fichier de clés des certificats racine**.
iii. Sélectionnez **DmgrDefaultRootStore**.
iv. Sélectionnez **Certificats personnels**.
v. Cliquez sur la case à cocher en regard d'un certificat dans le fichier de clés root. Indiquez le nom de fichier qualifié complet du certificat à extraire, par exemple : `/certificates/public.cer`.
vi. Dans une fenêtre de ligne de commande, exécutez la commande suivante : cd `/java_home/bin`
vii. Exécutez l'utilitaire keytool.

```
keytool -import -noprompt -alias "example alias" -keystore
/downloads/trustStore.jks  -file /certificates/public.cer -storepass
xc10pass -storetype jks
```

viii. Si vous avez d'autres certificats à importer, répétez les étapes d'extraction des certificats, puis relancez l'utilitaire keytool.

3. Téléchargez les informations de fichier de clés et de fichier de clés certifiées vers le dispositif. Dans l'interface utilisateur du dispositif, cliquez sur **Collectivité** > **Paramètres** > **Transport Layer Security (TLS)**. Si vous avez terminé les étapes de WebSphere Application Server, téléchargez le fichier `/downloads/trustStore.jks` mis à jour. Vous devez ensuite mettre à jour le mot de passe associé. Si vous utilisez le fichier de clés certifiées par défaut, le mot de passe est `xc10pass`.
4. Sélectionnez l'alias de certificat de la collectivité.
5. Définissez le type de transport. Choisissez l'un des paramètres de type de transport suivants :
   - **TLS pris en charge** : Les grilles de données communiquent à l'aide de TCP/IP, SSL ou TLS. L'interface utilisateur est accessible à l'aide de HTTP et HTTPS.
   - **TLS requis** : Les grilles de données communiquent à l'aide de SSL ou TLS uniquement. L'interface utilisateur est accessible à l'aide de HTTPS uniquement.
   - : Les grilles de données communiquent à l'aide de connexions non sécurisées. L'interface utilisateur est accessible à l'aide de HTTP et HTTPS.
6. Pour que le client envoie un certificat accrédité pour activer la communication, sélectionnez **Activer l'authentification du certificat client**.
7. Cliquez sur **Soumettre les paramètres TLS** pour enregistrer les modifications de la configuration.

Résultats

La collectivité doit redémarrer pour terminer l'application des modifications à la configuration TLS.

Certaines parties de l'interface utilisateur sont accessibles lorsque la collectivité redémarre. Si vous ne pouvez pas accéder à des parties de l'interface utilisateur, attendez un certain temps et renvoyez la demande. Le panneau des tâches indique certaines modifications automatiquement en affichant l'état d'aboutissement.

Si vous avez modifié l'alias de certificat utilisé par le dispositif, il peut être nécessaire de redémarrer le navigateur, de vous déconnecter et de vous reconnecter à l'interface utilisateur ou d'accréditer de nouveaux certificats à partir d'une invite de navigateur.

Si l'interface utilisateur semble indisponible lorsque l'authentification du client est activée, vérifiez que vous avez importé un certificat de client accrédité vers le navigateur. Si vous ne l'avez pas fait, vous ne pouvez pas accéder à l'interface utilisateur. Une fois connecté à l'interface utilisateur, la tâche indique l'aboutissement de la configuration TLS.

Que faire ensuite

Meilleures pratiques

- Pour éviter les avertissements de navigateur lorsque vous accédez à l'interface utilisateur à partir de dispositifs différents, incluez un caractère générique dans le nom usuel du certificat dans le fichier de clés certifiées. Chaque dispositif utilise le même certificat pour la configuration TLS que celle définie par l'alias de certificat. Par exemple, vous pouvez utiliser `*.mycompany.com` au lieu de `myhost.mycompany.com` pour que le certificat soit valide pour tous les hôtes du domaine `mycompany`.
- Vous pouvez utiliser une autorité de certificat (CA) privée pour signer le certificat associé à l'alias de certificat que vous avez choisi pour la configuration TLS. Ensuite, vous pouvez importer le certificat CA vers le navigateur et accréditer n'importe quelle collectivité avec un certificat signé par la CA privée sans afficher d'invite. L'utilisation d'une CA privée s'applique généralement uniquement pour l'accès à un intranet privé.

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Surveillance avec l'utilitaire xscmd

Référence associée:
Passerelle REST : Configuration de la sécurité
Fichier splicer.properties

Information associée:
Gestion des clés avec l'interface graphique IKEYMAN

Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance

Une grande partie des fonctions de sécurité offertes par WebSphere DataPower XC10 Appliance sont déjà intégrées dans le dispositif lors de sa fabrication. D'autres paramètres sont inclus pour proposer des options de sécurité supplémentaires pour votre environnement.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pour vous familiariser avec les fonctions de sécurité intégrées dans le dispositif, consultez la rubrique IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité.

Pourquoi et quand exécuter cette tâche

En supplément des fonctions de sécurité du dispositif, vous pouvez configurer plusieurs options disponibles pour contrôler le comportement de l'utilisateur.

Procédure

1. Accédez au panneau Paramètres. Pour gérer vos options de sécurité, accédez au panneau Paramètres à l'aide d'une des méthodes suivantes :
   - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
   - Dans la page Bienvenue, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Sécurité**.
3. Définissez vos autorisations de sécurité.
   a. Configurez la zone **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. La valeur par défaut de cette zone est *Désactivé*. Elle permet de spécifier si l'utilisateur est autorisé à créer son propre compte. Lorsque la valeur Activée est sélectionnée, un bouton **S'enregistrer** apparaît sur l'écran de connexion. Pour plus d'informations sur l'auto-enregistrement de l'utilisateur, reportez-vous à la rubrique Enregistrement d'un nouveau compte utilisateur.
   b. Configurez la zone *Permettre la réinitialisation du mot de passe à partir de la console série*. La valeur par défaut de cette zone est *Désactivé*.

      Désactivé : veillez à configurer un serveur SMTP et une adresse électronique pour l'utilisateur xcadmin. Ces configurations permettent de réinitialiser le mot de passe dans le cas où l'utilisateur xcadmin viendrait à perdre son mot de passe. Si cette zone est désactivée et que ces configurations ne sont pas effectuées, il est impossible de réinitialiser le mot de passe xcadmin. Dans ce cas, le dispositif devra être retourné à IBM pour réinitialisation.

      Activé : Vous pouvez réinitialiser le mot de passe de l'utilisateur xcadminà l'aide d'une connexion série sans autres données d'identification ni message SMTP. Si cette option est sélectionnée, le contrôle de l'accès physique à votre système WebSphere DataPower XC10 Appliance est encore renforcé par rapport à la normale. En effet, par le biais d'un accès physique à la machine, n'importe quel utilisateur peut obtenir un accès en tant qu'administrateur du dispositif.
4. Configurez votre dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP (Lightweight Directory Access Protocol). Pour plus d'informations sur la configuration du dispositif afin d'authentifier les utilisateurs avec un annuaire LDAP, voir Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP.

Résultats

A l'issue de ces étapes, vous aurez déterminé comment le dispositif gère certains scénarios affectant la sécurité et spécifié si une authentification externe est utilisée pour permettre l'accès à l'interface utilisateur.

Que faire ensuite

Configurez des utilisateurs et des groupes pour fournir l'accès à l'interface utilisateur. Vous utilisez également des utilisateurs et des groupes pour fournir l'accès aux grilles de données.

Rubrique parent : Sécurité

Concepts associés:

IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données

Référence associée:
Passerelle REST : Configuration de la sécurité

Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Si vous le souhaitez, vous pouvez utiliser un annuaire LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs sur votre système IBM® WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces opérations.

Pourquoi et quand exécuter cette tâche

L'utilisation d'un serveur LDAP pour authentifier les utilisateurs est facultative. L'attribut *nom d'utilisateur* est utilisé pour authentifier les utilisateurs d'IBM WebSphere DataPower XC10 Appliance dans l'annuaire LDAP. Les utilisateurs ne figurant pas dans l'annuaire LDAP ne peuvent pas être authentifiés.

2.5+ Vous pouvez configurer WebSphere DataPower XC10 Appliance pour qu'il procède à l'authentification LDAP via une connexion SSL. Pour ce faire, vous devez spécifier une URL LDAPS (Lightweight Directory Access Protocol over SSL) pour la connexion au serveur LDAP. Si vous utilisez LDAPS, le fichier de clés certifiées du dispositif doit être modifié pour qu'il accepte le certificat SSL du serveur LDAP. Si ce certificat a été émis par une autorité de certification, le certificat de signataire racine de cette autorité doit être ajouté au fichier de clés certifiées du dispositif. Si le certificat SSL du serveur LDAP est autosigné, il doit être ajouté au fichier de clés certifiées du dispositif.

2.5+ Lorsque l'authentification LDAP est configurée, tout ID utilisateur LDAP qui correspond au nom de domaine configuré de base et au filtre de recherche configuré pour les utilisateurs peut s'authentifier sur le dispositif. Cet utilisateur possède les droits et l'accès à la grille de données qui lui sont accordés ainsi que les droits d'accès et l'accès à la grille de données qui sont accordés aux groupes LDAP auxquels il appartient. Il n'est pas nécessaire d'ajouter l'utilisateur à la collectivité. Tous les utilisateurs LDAP ont également les droits et l'accès qui sont accordés au groupe Tout le monde. Vous pouvez ajouter un utilisateur à la collectivité afin que l'accès à la grille de données et les droits puissent être configurés pour cet utilisateur.

Lorsque l'authentification LDAP est configurée, vous ne pouvez ajouter que des groupes LDAP à la collectivité. L'accès et les droits peuvent être accordés aux groupes spécifiés.

Lorsque l'authentification LDAP est configurée, vous ne pouvez pas utiliser la console d'administration pour le dispositif ou des interfaces de programmation du dispositif pour ajouter ou supprimer des membres d'un groupe. L'appartenance à un groupe doit être gérée avec vos outils d'administration d'annuaire LDAP.

Pour les versions de IBM WebSphere DataPower XC10 Appliance antérieures à la version 2.5, seuls les utilisateurs qui sont spécifiquement ajoutés à la collectivité bénéficient de droits et d'accès. La prise en charge d'un LDAP généralisé est ajouté en version 2.5. Pour les versions antérieures à la version 2.5, le dispositif importe les appartenances de groupe de LDAP lorsque le groupe est ajouté à la collectivité. Le groupe est ensuite géré sur le dispositif, et l'appartenance des membres peut diverger de ce qui a été stocké dans LDAP. A partir de la version 2.5, si l'authentification LDAP est configurée, les appartenances à un groupe sont toujours résolues par l'interrogation du serveur LDAP.

Remarques sur la migration : Dans une collectivité qui inclut la version 2.5, ainsi que les dispositifs qui exécutent des versions de microprogramme antérieures, les utilisateurs qui sont dans LDAP sans être stockés dans la collectivité du dispositif ne peuvent pas accéder à des ressources restreintes sur des périphériques avec l'ancien microprogramme. Par conséquent, les clients peuvent uniquement utiliser les ID utilisateur qui sont ajoutés à la collectivité du dispositif jusqu'à ce que tous les périphériques soient mis à niveau.

Lorsqu'une collectivité comprend des membres qui exécutent des versions de microprogramme antérieures à la version 2.5, il est possible que les appartenances de groupe qui sont stockées sur d'anciens dispositifs divergent de ce qui est stocké dans LDAP. Ces incohérences peuvent provoquer des problèmes. Par exemple, si un ID utilisateur est dans un groupe qui est stocké sur un dispositif plus ancien et si les autorisations et les accès sont associés à ce groupe mais que ce groupe n'existe pas dans LDAP, ou si l'appartenance au groupe qui est stockée sur le dispositif diffère de ce qui figure dans LDAP, l'ID utilisateur ne pourra peut-être pas accéder à des ressources restreintes sur le nouveau dispositif. Ce comportement est dû au fait que le dispositif en version 2.5 vérifie LDAP directement, et non pas n'importe quelle version locale de l'appartenance à un groupe. Lorsque vous effectuez une migration, vérifiez que les ID utilisateur qui sont utilisés pour accéder aux données du dispositif disposent des droits et des accès nécessaires. Vérifiez également que ces ID utilisateur représentent des membres des groupes LDAP qui disposent des autorisations requises.

Procédure

1. Accédez au panneau **Paramètres**. Procédez de l'une des manières suivantes :

- Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
- Dans la page Bienvenue, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.

2. Développez l'entrée **Sécurité**.
3. Configurez votre dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP.

   a. Pour activer ce mécanisme, cochez la case en regard de l'option **Activer l'authentification LDAP**. Par défaut, la case **Activer l'authentification LDAP** n'est pas sélectionnée. La sélection de cette case indique à WebSphere DataPower XC10 Appliance d'utiliser le serveur LDAP pour authentifier les utilisateurs lors de leur connexion.

   b. Entrez l'URL du fournisseur JNDI. Exemple pour un annuaire LDAP non SSL :

      ```
      ldap://mycompany.com:389/
      ```

      ou

      ```
      ldap://mycompany.com/
      ```

      Si aucun port n'est spécifié de manière explicite, le numéro de port par défaut est 389. Exemple pour un annuaire LDAP SSL :

      ```
      ldaps://mycompany.com:636/
      ```

      ou

      ```
      ldaps://mycompany.com/
      ```

      Si aucun port n'est spécifié de manière explicite, le numéro de port par défaut est 636.

   c. Entrez le nom distinctif JNDI de base (utilisateurs). Exemple :

      ```
      CN=users,DC=mycompany,DC=com
      ```

   d. Entrez le nom distinctif JNDI de base (groupes). Exemple :

      ```
      DC=mycompany,DC=com
      ```

   e. Entrez le filtre de recherche (utilisateurs). Exemple :

      ```
      (&(sAMAccountName={0})(objectcategory=user)) ou uid={0}
      ```

      Remarque : L'ID utilisateur est incorporé dans l'espace réservé "{0}". "{0}" est remplacé par l'ID utilisateur de connexion saisi dans l'écran de connexion.

   f. Entrez l'authentification de sécurité JNDI. Cette zone est facultative sauf si votre serveur LDAP n'autorise pas les requêtes LDAP anonymes. Exemple :

      ```
      CN=Administrator,CN=users,DC=mycompany,DC=com
      ```

   g. Entrez le mot de passe. Cette zone correspondant aux données d'identification de sécurité JNDI est facultative, sauf si votre serveur LDAP n'autorise pas les requêtes LDAP anonymes.

4. Testez les paramètres d'authentification LDAP que vous avez définis. Vous pouvez tester les paramètres que vous avez définis pour configurer l'authentification avec le serveur LDAP. Cette section vous permet d'exécuter des requêtes LDAP afin de rechercher des utilisateurs ou des groupes.

   a. Cliquez sur **Test des paramètres d'authentification LDAP** pour développer cette section.

   b. Pour tester un nom d'utilisateur, entrez un nom d'utilisateur dans la zone Utilisateur LDAP, puis cliquez sur le bouton **Test de requête LDAP**. Exemple :

      ```
      utilisateur_test@us.ibm.com
      ```

      Si la requête aboutit, le message suivant s'affiche : *Nom distinctif d'utilisateur LDAP trouvé : <informations utilisateur>*. Si la requête échoue, un message d'erreur s'affiche.

   c. Pour tester un nom de groupe, entrez un nom de groupe dans la zone Groupe LDAP, puis cliquez sur le bouton **Test de requête LDAP** associé. Exemple :

```
Groupe test
```

Si la requête aboutit, le message suivant s'affiche : *Nom distinctif de groupe LDAP trouvé : <informations utilisateur>*. Si la requête échoue, un message d'erreur s'affiche.

Résultats

Vous avez défini un annuaire LDAP pour authentifier les utilisateurs externes qui accèdent à l'interface utilisateur.

Que faire ensuite

La maîtrise du contrôle d'accès des utilisateurs aux différentes zones de votre environnement constitue un élément important de votre solution de sécurité. Pour plus d'informations sur la gestion des utilisateurs et des groupes et de leurs autorisations, reportez-vous à la rubrique Gestion des utilisateurs et des groupes.

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données

Référence associée:
Passerelle REST : Configuration de la sécurité

Gestion des utilisateurs et des groupes

Des utilisateurs et des groupes sont fournis pour gérer les accès à votre dispositif WebSphere DataPower XC10 Appliance par chaque individu. Vous pouvez utiliser des groupes d'utilisateurs pour appliquer des autorisations à des groupes d'utilisateurs.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Vous pouvez gérer vos utilisateurs et vos groupes d'utilisateurs à partir de l'interface utilisateur de WebSphere DataPower XC10 Appliance .

Création d'un utilisateur
Un nom d'utilisateur et un mot de passe sont requis pour se connecter à l'interface utilisateur. Procédez comme suit pour créer de nouveaux comptes permettant à leurs utilisateurs d'accéder et de gérer votre WebSphere DataPower XC10 Appliance.

Gestion des utilisateurs
Une fois un utilisateur créé, vous devez le modifier manuellement dans le cas où d'autres autorisations sont nécessaires. Vous pouvez également suivre ces étapes pour modifier un utilisateur si l'information a changé.

Suppression d'un utilisateur
Un nom d'utilisateur et un mot de passe sont requis pour se connecter à interface utilisateur. Lorsque vous n'avez plus besoin d'un utilisateur, vous pouvez le supprimer de WebSphere DataPower XC10 Appliance.

Enregistrement d'un nouveau compte utilisateur
Vous pouvez créer votre propre compte utilisateur lorsque l'administrateur active l'option **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. Utilisez cette tâche pour créer un compte utilisateur à partir de l'écran de connexion.

Création d'un groupe d'utilisateurs
Vous pouvez créer des groupes d'utilisateurs afin de mieux gérer l'accès de vos utilisateurs à des ressources WebSphere DataPower XC10 Appliance spécifiques.

Gestion des groupes d'utilisateurs
Lorsque vous créez un groupe d'utilisateurs, ce dernier ne dispose pas de membres. Vous devez ajouter manuellement les utilisateurs au groupe si l'authentification LDAP n'est pas activée.

Suppression d'un groupe d'utilisateurs
Vous pouvez supprimer un groupe d'utilisateurs de WebSphere DataPower XC10 Appliance si le groupe d'utilisateurs est devenu inutile.

Droits utilisateur
Les droits utilisateur permettent de déterminer quels panneaux sont visibles par chaque utilisateur et les autorisations d'accès dont il dispose pour un objet spécifique.

Rubrique parent : Configuration de votre dispositif
Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Sécurité
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Création d'un utilisateur

Un nom d'utilisateur et un mot de passe sont requis pour se connecter à l'interface utilisateur. Procédez comme suit pour créer de nouveaux comptes permettant à leurs utilisateurs d'accéder et de gérer votre WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces opérations. Si vous utilisez LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs, l'utilisateur en cours d'enregistrement doit préalablement exister dans le référentiel LDAP. Pour plus d'informations sur la création d'une configuration LDAP, reportez-vous à la rubrique Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP .

Pourquoi et quand exécuter cette tâche

Procédez comme suit pour créer un utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance :

Procédure

1. Accédez au panneau **Utilisateurs**.
   - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance , accédez à **Collectivité** > **Utilisateurs**.
   - Dans le panneau d'**accueil**, cliquez sur le lien **Créer des utilisateurs** dans la section **Etape 1 : configurez l'appareil**.
2. Cliquez sur l'icône Ajouter ( ) pour lancer l'ajout d'un nouvel utilisateur.
3. Entrez un ID dans la zone **Nom d'utilisateur**. La valeur de cette zone peut comporter jusqu'à 64 caractères et ne doit pas être vide. Vous pouvez utiliser tous les caractères alphanumériques et la plupart des caractères spéciaux. Vous ne pouvez pas utiliser d'espaces ni les caractères spéciaux suivants : < #. Cette zone ne peut plus être modifiée après la création de l'utilisateur. Si vous utilisez LDAP, l'utilisateur en cours d'enregistrement doit préalablement exister dans le référentiel LDAP.
4. Facultatif : Entrez le nom de l'utilisateur dans la zone **Nom complet**. Cette zone est utilisée à des fins d'affichage dans l'interface utilisateur. Si vous n'entrez pas de valeur dans cette zone, le nom d'utilisateur s'affiche. Une fois l'utilisateur créé, seul l'utilisateur peut modifier la zone. L'administrateur ne peut pas changer la valeur de la zone après la création de l'utilisateur.
5. Entrez le mot de passe de l'utilisateur dans la zone **Mot de passe**. Le mot de passe peut contenir les mêmes caractères disponibles pour la zone **Nom d'utilisateur**. Si le protocole SMTP (Simple Mail Transfer Protocol) est activé, vous pouvez ne pas renseigner la zone du mot de passe lors de la création d'un utilisateur ; un mot de passe est créé automatiquement. Si l'authentification LDAP est activée, la zone **Mot de passe** n'est pas affichée puisque le mot de passe de l'annuaire LDAP est utilisé pour l'authentification. Entrez le même mot de passe de l'utilisateur dans la zone **Vérification du mot de passe**.
6. Entrez une adresse électronique valide pour l'utilisateur dans la zone **Adresse électronique**. Cette zone spécifie l'adresse électronique utilisée pour fournir un nouveau mot de passe si l'utilisateur a oublié sont mot de passe et d'autres notifications. L'adresse électronique est nécessaire lorsqu'un serveur SMTP (Simple Mail Transfer Protocol) est utilisé. Voir Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance pour plus d'informations.
7. Cliquez sur **OK**.

Résultats

Vous disposez d'un nouveau compte utilisateur que vous pouvez utiliser pour vous connecter à l'interface utilisateur.

Que faire ensuite

A sa création initiale, l'utilisateur ne dispose que des autorisations par défaut. Si vous désirez lui accorder des autorisations supplémentaires, consultez la rubrique Gestion des utilisateurs pour les instructions correspondantes. Pour ajouter le nouvel utilisateur à un groupe d'utilisateurs, reportez-vous à la rubrique Gestion des groupes d'utilisateurs.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:

Gestion des utilisateurs

Une fois un utilisateur créé, vous devez le modifier manuellement dans le cas où d'autres autorisations sont nécessaires. Vous pouvez également suivre ces étapes pour modifier un utilisateur si l'information a changé.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Lorsque vous créez un utilisateur, celui-ci possède les autorisations par défaut. Si ce compte utilisateur a besoin de permissions supplémentaires, vous devez les lui accorder manuellement après sa création initiale. Si un compte utilisateur a été créé via la fonction d'auto-enregistrement, seul un sous-ensemble des informations sur l'utilisateur est disponible. les informations complémentaires doivent être ajoutées par un utilisateur avec droits d'administration sur le dispositif. Procédez comme suit pour modifier un utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez au panneau **Utilisateurs**.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, accédez à **Collectivité** > **Utilisateurs**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Créer des utilisateurs** dans la section **Etape 1 : configurez l'appareil**.
2. Cliquez sur le ***nom d'utilisateur*** de l'utilisateur que vous désirez modifier. Le nom d'affichage et le nom d'utilisateur ne peuvent pas être modifiés une fois que l'utilisateur a été créé.
3. Vous pouvez éditer le mot de passe et l'adresse électronique de l'utilisateur. Pour modifier le mot de passe, cliquez sur le bouton **[éditer]** de la zone. Entrez un nouveau mot de passe pour modifier le mot de passe.
4. Visualisez l'activité de l'utilisateur. Vous pouvez visualiser les activités suivantes de l'utilisateur à partir de l'écran du compte utilisateur :
    - **Statut actuel** : Cette zone affiche le statut de l'utilisateur. La liste suivante répertorie les statuts possibles :
        - : utilisateur actif au cours des cinq dernières minutes
        - : utilisateur inactif depuis plus de cinq minutes
        - : utilisateur non connecté
    - **Groupes d'utilisateurs** : Cette zone répertorie tous les groupes d'utilisateurs dont l'utilisateur fait partie.

      Entrez un nom de groupe pour ajouter un utilisateur à ce groupe. Quand vous entrez un nom de groupe d'utilisateurs, une liste de groupes d'utilisateurs proches de votre saisie apparaît. Vous devez cliquer sur le nom du groupe d'utilisateurs approprié pour ajouter l'utilisateur à ce groupe. Il ne suffit pas de taper le nom du groupe d'utilisateurs pour y ajouter l'utilisateur. Quand vous ajoutez un utilisateur à un groupe d'utilisateurs, l'utilisateur reçoit les droits d'accès affectés à ce groupe d'utilisateurs. Le niveau de droits d'accès précédent de l'utilisateur n'est pas conservé. Si vous désirez supprimer un utilisateur dans un groupe d'utilisateurs, cliquez sur le lien **[supprimer]** situé en regard du groupe d'utilisateurs dont vous voulez exclure l'utilisateur. Si vous supprimez un utilisateur de tous les groupes d'utilisateurs (en plus du groupe Tout le monde), l'utilisateur conserve les droits d'accès affectés au dernier groupe dont il a été exclu.
5. Modifiez les autorisations de cet utilisateur. Vous pouvez sélectionner ou désélectionner ces autorisations pour contrôler le niveau d'accès affecté à un utilisateur. Vous ne pouvez pas modifier les autorisations d'un utilisateur s'il fait partie d'un groupe, hormis le groupe Tout le monde. Si un utilisateur est membre d'un groupe, il possède les droits d'accès affectés à ce groupe. Si un utilisateur est membre de plusieurs groupes, il possède tous les droits d'accès affectés à tous ces groupes. Quand vous modifiez les droits d'accès affectés à un groupe, la modification est propagée à tous les membres de ce groupe. Les autorisations suivantes sont disponibles pour chaque utilisateur :
    - Administration de dispositif
    - Contrôle du dispositif
    - Création d'une grille de données

    Pour plus d'informations sur ces paramètres d'autorisations, reportez-vous à la rubrique Droits utilisateur.

Résultats

Vous avez modifié un compte utilisateur.

Que faire ensuite

Après avoir modifié l'utilisateur, vous pouvez l'ajouter à un groupe d'utilisateurs. Reportez-vous aux rubriques Création d'un groupe d'utilisateurs et Gestion des groupes d'utilisateurs pour plus d'informations sur la création d'un groupe d'utilisateurs et sur l'ajout d'utilisateurs à des groupes. Vous pouvez ajouter une couche de sécurité à votre dispositif en utilisant un serveur LDAP (Lightweight Directory Access Protocol) pour les authentifications. Pour plus d'informations sur la sécurisation de votre dispositif à l'aide d'un serveur LDAP, reportez-vous à la rubrique Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Suppression d'un utilisateur

Un nom d'utilisateur et un mot de passe sont requis pour se connecter à interface utilisateur. Lorsque vous n'avez plus besoin d'un utilisateur, vous pouvez le supprimer de WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Lorsque vous supprimez un utilisateur, toutes les ressources dont il était propriétaire vous sont automatiquement transférées. Procédez comme suit pour supprimer du dispositif un compte utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez au panneau **Utilisateurs**.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance , accédez à **Collectivité** > **Utilisateurs**.
    - Dans la page d'**accueil**, cliquez sur le lien **Créer des utilisateurs** dans la section **Etape 1 : configurez l'appareil**.
2. Cliquez sur le ***<nom_d'utilisateur>*** de celui que vous désirez supprimer.

    Remarque : Il est impossible de supprimer le compte de l'administrateur (xcadmin).

3. Cliquez sur l'icône de suppression ( ) pour lancer la suppression. Un message s'affiche pour vous inviter à confirmer la suppression définitive de cet utilisateur.
4. Cliquez sur **OK**.

Résultats

Vous avez supprimé un compte utilisateur du dispositif.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Enregistrement d'un nouveau compte utilisateur

Vous pouvez créer votre propre compte utilisateur lorsque l'administrateur active l'option **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. Utilisez cette tâche pour créer un compte utilisateur à partir de l'écran de connexion.

Avant de commencer

Pour enregistrer un nouveau compte utilisateur, sélectionnez *Activé* dans la zone **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. Pour plus d'informations sur l'activation de l'auto-enregistrement des utilisateurs, reportez-vous à la rubrique Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez à l'écran de connexion d'WebSphere DataPower XC10 Appliance.
2. Cliquez sur le bouton **S'enregistrer...** pour lancer la procédure de création d'un compte utilisateur.
3. Entrez un ID de connexion dans la zone **Nom d'utilisateur**. La valeur saisie dans cette zone est utilisée à la fois comme nom d'utilisateur et comme nom d'affichage. Cette zone peut contenir une valeur comportant jusqu'à 64 caractères. Tous les caractères alphanumériques sont admis. Les caractères spéciaux suivants peuvent également être utilisés : *!@#%^*&-+=*

   Important : Si LDAP (Lightweight Directory Access Protocol) est utilisé pour authentifier les utilisateurs, l'utilisateur en cours d'enregistrement doit préalablement exister dans le référentiel LDAP. Pour plus d'informations sur la création d'une configuration LDAP, reportez-vous à la rubrique Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP .

4. Entrez le mot de passe de l'utilisateur dans la zone **Mot de passe**. Vous pouvez utilisez pour le mot de passe les mêmes caractères que pour la zone Nom d'utilisateur. La zone **Mot de passe** doit être définie si un serveur SMTP (Simple Mail Transfer Protocol) ou LDAP est défini. Si SMTP est activé, vous pouvez entrer un mot de passe ou laissez la zone vide et faire envoyer un mot de passe généré à votre adresse électronique. Si LDAP est utilisé pour authentifier les utilisateurs le mot de passe LDAP existant est utilisé et vous n'avez pas à entrer de mot de passe.
5. Rentrez le même mot de passe de l'utilisateur dans la zone **Vérification du mot de passe**. La valeur saisie dans cette zone doit être identique à celle spécifiée dans la zone **Mot de passe**. Si le contenu de ces zones ne correspond pas, un message d'erreur est affiché lorsque vous cliquez sur **Enregistrement** et vous devez corriger cette erreur avant de pouvoir créer l'utilisateur.
6. Entrez une adresse électronique dans la zone **Adresse électronique**. L'adresse électronique est nécessaire lorsqu'un serveur SMTP (Simple Mail Transfer Protocol) est utilisé. Voir Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance pour plus d'informations.
7. Cliquez sur le bouton **S'enregistrer** pour terminer la procédure d'enregistrement.

Résultats

A l'issue de ces étapes, vous aurez enregistré un compte utilisateur que vous pourrez utiliser pour vous connecter à l'interface utilisateur. Par défaut, seules les autorisations de contrôle du dispositif sont affectées.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Création d'un groupe d'utilisateurs

Vous pouvez créer des groupes d'utilisateurs afin de mieux gérer l'accès de vos utilisateurs à des ressources WebSphere DataPower XC10 Appliance spécifiques.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Créez des groupes d'utilisateurs pour affecter rapidement à une collection d'utilisateurs un accès à une ressource ou un groupe de ressources. Les groupes d'utilisateurs sont vides à leur création initiale. Vous devez ajouter manuellement des utilisateurs à chaque nouveau groupe d'utilisateurs. Procédez comme suit pour créer un utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez à **Collectivité** > **Groupes d'utilisateurs**.
2. Cliquez sur l'icône Ajouter ( ) pour créer un groupe.
3. Entrez un nom dans la zone Nom de groupe. La valeur de cette zone peut comporter jusqu'à 64 caractères et ne doit pas être vide. Tous les caractères alphanumériques peuvent être utilisés, ainsi que les caractères spéciaux suivants : *!@#%^*&-+=* . Lorsque l'authentification LDAP est configurée, vous ne pouvez ajouter que des groupes qui existent dans LDAP pour la collectivité.
4. Entrez éventuellement des informations supplémentaires dans la zone Description. Cette zone peut être utilisée pour inclure des informations supplémentaires sur le groupe d'utilisateurs.
5. Cliquez sur **OK**.

Résultats

A l'issue de ces étapes, vous disposerez d'un nouveau groupe d'utilisateurs pour vous faciliter la gestion des autorisations de vos utilisateurs WebSphere DataPower XC10 Appliance.

Que faire ensuite

Vous pouvez ajouter des utilisateurs au groupe d'utilisateurs que vous avez créé. Reportez-vous à la rubrique Gestion des groupes d'utilisateurs pour les instructions détaillées sur l'ajout d'utilisateurs à un groupe.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Gestion des groupes d'utilisateurs

Lorsque vous créez un groupe d'utilisateurs, ce dernier ne dispose pas de membres. Vous devez ajouter manuellement les utilisateurs au groupe si l'authentification LDAP n'est pas activée.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Suivez la procédure ci-dessous pour ajouter ou supprimer un utilisateur d'un groupe d'utilisateurs à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez à **Collectivité** > **Groupes d'utilisateurs**.
2. Cliquez sur un **<nom de groupe>** pour sélectionner un groupe à modifier.
3. Si vous voulez modifier la description du groupe d'utilisateurs, cliquez sur la description existante et entrez les modifications à effectuer.
4. Si vous voulez ajouter un utilisateur au groupe, tapez l'utilisateur à ajouter, puis cliquez sur le **<nom d'utilisateur>**

   A mesure que vous entrez le nom de l'utilisateur, une liste des utilisateurs correspondant à votre saisie s'affiche. Vous devez cliquez sur le nom de l'utilisateur approprié pour l'ajouter au groupe. La seule saisie du nom de l'utilisateur ne l'ajoute pas pour autant au groupe. Quand vous ajoutez un utilisateur à un groupe d'utilisateurs, l'utilisateur reçoit les droits d'accès affectés à ce groupe d'utilisateurs. Le niveau de droits d'accès précédent de l'utilisateur n'est pas conservé.

   Remarque : Si vous avez activé l'authentification LDAP, vous ne pouvez pas modifier la composition d'un groupe dans WebSphere DataPower XC10 Appliance.

5. Modifiez les droits d'accès affectés au groupe.

   Vous pouvez affecter les droits d'accès suivants à un groupe d'utilisateurs :
   - Administration de dispositif
   - Contrôle du dispositif
   - Création de cache de données

   Pour plus d'informations sur ces paramètres d'autorisations, reportez-vous à la rubrique Droits utilisateur.

6. Si vous désirez supprimer un utilisateur d'un groupe, cliquez sur le lien **[supprimer]** en regard de l'utilisateur concerné. Aucune confirmation n'étant nécessaire pour supprimer l'utilisateur, gérez le groupe d'utilisateurs avec précaution. Si vous supprimez un utilisateur de tous les groupes, autre que le groupe Tout le monde, l'utilisateur conserve les autorisations affectées au dernier groupe dont il est supprimé.

Résultats

Vous avez terminé la modification du groupe d'utilisateurs.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Suppression d'un groupe d'utilisateurs

Suppression d'un groupe d'utilisateurs

Vous pouvez supprimer un groupe d'utilisateurs de WebSphere DataPower XC10 Appliance si le groupe d'utilisateurs est devenu inutile.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Procédez comme suit pour supprimer un groupe d'utilisateurs du dispositif en utilisant l'interface utilisateur WebSphere DataPower XC10 Appliance .

Procédure

1. Cliquez sur **Collectivité** > **Groupes d'utilisateurs**.
2. Cliquez sur le ***<nom_groupe_utilisateurs>*** pour sélectionner celui que vous comptez supprimer.

   Remarque : Le groupe Tout le monde ne peut pas être supprimé.

3. Cliquez sur l'icône de suppression ( ) pour supprimer le groupe.
4. Cliquez sur **OK** pour confirmer la suppression du groupe d'utilisateurs sélectionnés.

Résultats

Chaque membre du groupe d'utilisateurs est supprimé du groupe et le groupe lui-même est supprimé.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs

Droits utilisateur

Les droits utilisateur permettent de déterminer quels panneaux sont visibles par chaque utilisateur et les autorisations d'accès dont il dispose pour un objet spécifique.

Les droits attribués aux utilisateurs définissent les tâches d'administration WebSphere DataPower XC10 Appliance qu'ils sont habilités à effectuer. Ces droits déterminent les pages d'administration qui sont affichées. En outre, le contenu de la page **Bienvenue** est généré dynamiquement afin d'afficher aux utilisateurs des contenus différents en fonction de leur niveau d'accès. Lorsque les utilisateurs s'enregistrent pour la première fois, ils sont autorisés à contrôler le dispositif. Un administrateur doit affecter des droits de création de grille de données ou d'administration de dispositif.

Tableau 1. Ecrans visibles pour chaque niveau de droits

	Droits de contrôle du dispositif	**Droits de création de grille de données**	**Droits d'administration du dispositif**
Page Bienvenue	Oui	Oui	Oui
Création de grilles de données	Non	Oui	Oui
Suppression de grilles de données	Non	Oui	Oui
Affichage du menu de contrôle	Oui	Non	Oui
Affichage des tâches	Oui	Oui	Oui
2.5+ Suppression des tâches terminées	Non	Non	Oui
Affichage et création des collectivités et des zones	Non	Non	Oui
Modification des paramètres du dispositif	Non	Non	Oui
Modification des utilisateurs et des groupes	Non	Non	Oui
2.5+ Visualisation de la gestion des données	Oui	Non	Oui
2.5+ Visualisation de Message Center	Oui	Non	Oui

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Mot de passe xcadmin

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Passerelle REST : Configuration de la sécurité

Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

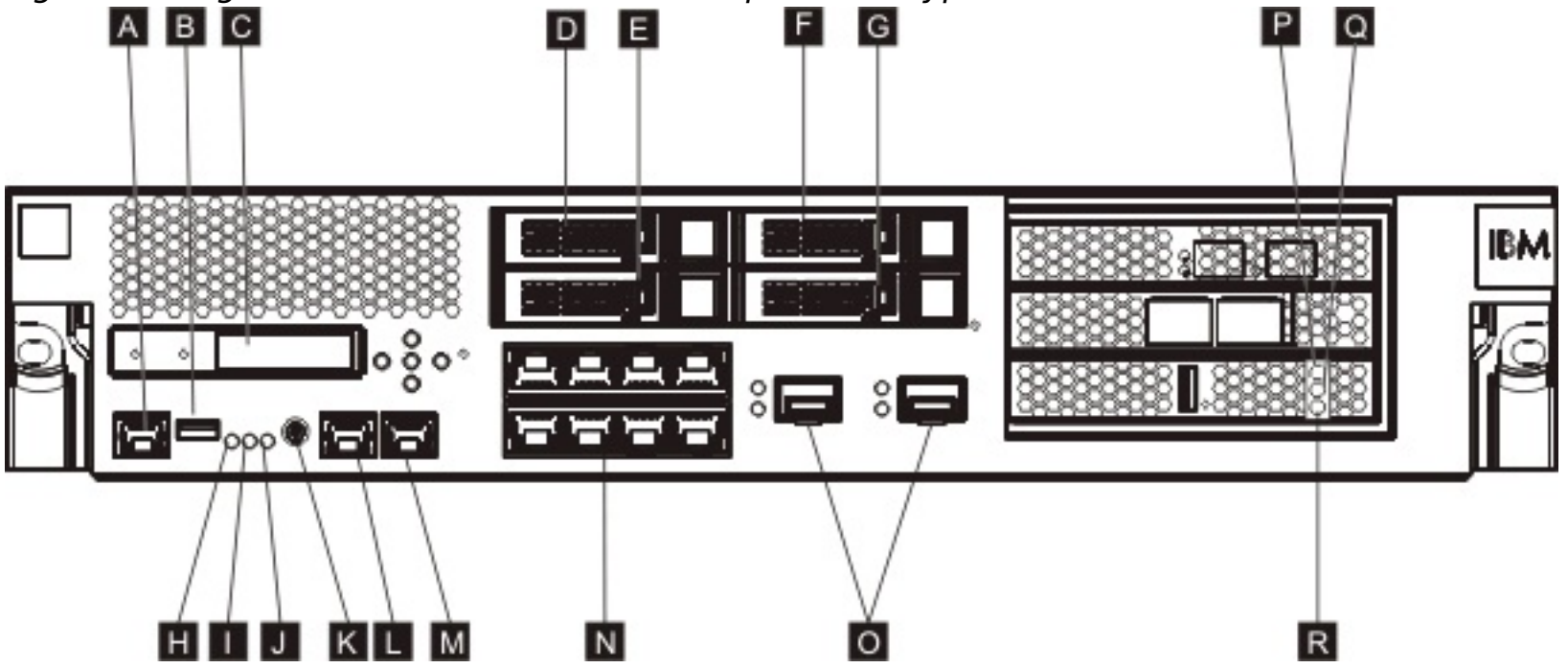
Lors de la connexion série initiale, vous avez configuré l'interface Ethernet *mgmt* afin de connecter le dispositif à votre réseau. Vous pouvez définir des ports Ethernet privés supplémentaires dans l'interface utilisateur.

Avant de commencer

Vous devez disposer des droits d'accès d'administrateur du dispositif.

Pourquoi et quand exécuter cette tâche

Figure 1. Diagramme de la face avant du dispositif de type 7199-92x



- mgmt0 (L)
- mgmt1(M)
- Gauche - Modules Ethernet 1 gigabit eth0-eth7 (N)
- Droite - Modules Ethernet 10 gigabits eth8-eth9 (O)

Pour le dispositif de type 7199-92x, utilisez les ports Ethernet 1 gigabit ou 10 gigabits pour votre grille de données. Vous ne pouvez pas utiliser à la fois des ports 1 gigabit et des ports 10 gigabits. Vous ne pouvez pas changer de type de port après la configuration initiale. Connectez le port de gestion à `MGMT0` (L).

Si vous modifiez les interfaces Ethernet pour un dispositif autonome, vous devez effacer la configuration et redémarrer le dispositif après avoir modifié les paramètres. Si le dispositif fait partie d'une collectivité, vous ne pouvez pas mettre à jour les interfaces Ethernet.

Procédure

1. Modifiez les interfaces Ethernet sur un dispositif autonome. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.
2. Développez l'entrée **Interfaces Ethernet**.
3. Activez ou désactivez une interface Ethernet en cochant ou en désélectionnant la case **Activée**. L'interface *mgmt* ne peut pas être désactivée.
4. Modifiez le **masque/l'adresse IP**. Entrez l'adresse IP et le masque de sous-réseau sous le format suivant *<adresse_ip>/<masque_sous-réseau>*. Le masque de sous-réseau doit être indiqué en respectant la notation CIDR (Classless Inter-Domain Routing). Exemple : *255.255.255.0* en notation longue devient *24* en notation CIDR.
5. Modifiez la **passerelle par défaut**. Le dispositif utilise un routage basé sur la source et non sur la destination. Un paquet est envoyé vers sa destination via l'interface par laquelle il est arrivé. Chaque interface dispose de sa propre table de routage, qui est distincte de celle des autres interfaces. Pour chaque interface qui doit atteindre des destinations au-delà du sous-réseau local, indiquez un chemin par défaut accessible directement depuis cette interface.
6. Indiquez si l'adresse IP fournie est une **Adresse IP privée**.
7. Modifiez l'**unité de transmission maximale (MTU)**. Cette zone spécifie la taille maximale, en octets, d'une unité de données du protocole lors d'une communication via une interface Ethernet. Sa valeur par défaut est de 1500 octets, ce qui est également la valeur maximale autorisée pour cette zone.

8. Modifiez le **mode**. Les modes suivants sont disponibles pour vos interfaces Ethernet :
    - AUTO (Automatique)
    - 10baseT-HD
    - 10baseT-FD
    - 100baseTx-HD
    - 100baseTx-FD
    - 1000baseTx-FD
9. Effacez la configuration, puis redémarrez le dispositif en cochant la case **Activée**. Cette opération efface la configuration. Le redémarrage est nécessaire pour que les processus WebSphere DataPower XC10 Appliance se lient à l'interface Ethernet.

Que faire ensuite

Pour surveiller l'état de vos interfaces Ethernet, voir Surveillance du statut des interfaces Ethernet à l'aide d'une connexion série.

Ajout d'une interface d'agrégation
Vous pouvez configurer une interface d'agrégation dans WebSphere DataPower XC10 Appliance.

Modification d'une interface d'agrégation
Vous pouvez ajouter ou supprimer des membres d'une interface d'agrégation et modifier des propriétés telles que l'adresse IP, la passerelle par défaut ou l'unité de transmission maximale (MTU). Vous pouvez également modifier la règle d'agrégation.

Suppression d'une interface d'agrégation
Vous pouvez supprimer une interface d'agrégation.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Ajout d'une interface d'agrégation

Vous pouvez configurer une interface d'agrégation dans WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de WebSphere DataPower XC10 Appliance pour pouvoir réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Vous pouvez lier plusieurs interfaces réseau à une interface d'agrégation unique. Une interface d'agrégation est constituée d'une ou plusieurs interfaces Ethernet servant d'unité logique unique. Cette fonction permet de répartir le trafic sur votre réseau. Vous pouvez utiliser l'interface de ligne de commande ou la console Web pour créer une interface d'agrégation. Il est recommandé d'utiliser l'interface de ligne de commande. Avant chaque opération, vous devez vous connecter au dispositif. Après l'opération, exécutez la commande **`clear-all`** et redémarrez le dispositif.

L'agrégation de liaison entre les commutateurs augmente la connectivité et la bande passante et renforce la redondance.

- Bande passante accrue

  Plusieurs interfaces Ethernet sont regroupées en une seule liaison logique, sous forme d'interface d'agrégation, ce qui augmente la bande passante disponible pour le dispositif.

  A faire : Avec le dispositif, vous pouvez activer des ports de 1 Gbps ou de 10 Gbps. Par conséquent, si vous activez des ports Ethernet de 10 Gbps, vous pouvez ensuite agréger également des ports de 10 Gbps.

- Basculement automatique

  Le trafic sur un port Ethernet défaillant d'une agrégation est transféré vers un port Ethernet actif. L'utilisation d'un commutateur unique offre une protection contre les incidents de port. L'utilisation de commutateurs redondants offre une protection contre les incidents de commutateur réseau.

- Equilibrage de charge

  Vous pouvez sélectionner des règles d'équilibrage de charge pour répartir le trafic entrant et sortant.

- Prise en charge de la redondance

  Deux systèmes peuvent être connectés par plusieurs liaisons dans des configurations d'agrégation différentes.

- Administration améliorée

  En tant qu'administrateur, vous pouvez gérer plusieurs interfaces Ethernet sous la forme d'une unité d'interface d'agrégation unique.

Procédure

- Ajout d'une interface d'agrégation à l'aide de l'interface de ligne de commande. Aidez-vous de l'exemple suivant pour ajouter l'interface d'agrégation :

```
Console> create aggregate-interface myAgg
Console aggregate-interface:myAgg> member eth3 eth4
Console aggregate-interface:myAgg> ip
CWZBR02205I: Entering "ip" mode
Console aggregate-interface:myAgg ip> address 10.5.5.5/24
Console aggregate-interface:myAgg ip> exit
Console aggregate-interface:myAgg> primary-member eth3
Console aggregate-interface:myAgg> aggregation-policy balance-tlb
Console aggregate-interface:myAgg> show
aggregate-interface myAgg:
name "myAgg"
AdminState "Enabled"
use-arp "true"
mtu "1500"
ip
  use-dhcp "false"
  address "10.5.5.5/24"
```

```
  use-slaac "false"
  dad-transmits "1"
  dad-retransmit-timer "1000"
end ip
aggregation-policy "balance-tlb"
lacp-selection-logic "stable"
transmit-hash-policy "layer2"
member "eth3" "eth4"
primary-member "eth3"

Console aggregate-interface:myAgg> exit
```

- Ajout d'une interface d'agrégation à l'aide de la console Web. Affichez le panneau Paramètres de l'une des manières suivantes :
  - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
  - Dans le panneau d'accueil, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.

1. Développez l'entrée **Interfaces d'agrégation**.
2. Cliquez sur **Ajouter une nouvelle interface d'agrégation**.
3. Complétez le formulaire pour décrire l'interface d'agrégation que vous souhaitez ajouter.

   Nom

   Indique le nom de l'interface.

   Adresse/masque IP

   Indique l'adresse IP que vous souhaitez affecter à l'interface d'agrégation. Entrez l'adresse IP et le masque de sous-réseau sous le format suivant *<adresse_ip>/<masque_sous-réseau>*.

   Conseil : Les adresses IPv4 et IPv6 sont toutes deux affectées à l'aide de cette propriété. Les adresses en double ne sont pas admises.

   Règle d'agrégation

   active-backup
   Indique la règle active-backup pour la haute disponibilité. A l'aide de cette règle, une seule interface Ethernet, en tant que membre d'une interface d'agrégation, est activée à la fois. Si cette interface Ethernet échoue, un autre membre reprend le traitement.
   LACP
   Indique la règle LACP (Link Aggregation Control Protocol) pour la haute disponibilité et la bande passante. Vous ne pouvez utiliser la règle LACP que lorsque son mode n'est pas paramétré sur 'OFF'. Par défaut, la logique de sélection est `Stable` et le mode de hachage de transmission est `layer2`.
   balance-tlb
   Indique la règle balance-tlb pour l'équilibrage de charge et la haute disponibilité. Cette règle distribue le trafic sortant en fonction de la charge en cours de chaque membre. Le trafic entrant est transmis à l'interface Ethernet sélectionnée comme membre principal. Si l'interface Ethernet de réception échoue, un autre membre reprend le traitement.

   Membres

   Indique l'interface Ethernet que vous souhaitez ajouter à l'interface d'agrégation. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre principal.

   Restriction : Une interface Ethernet ne peut pas être membre de plusieurs interfaces d'agrégation.

   Important : Assurez-vous que l'interface Ethernet est désactivée avant de l'ajouter à l'interface d'agrégation. Pour désactiver l'interface Ethernet, vous devez désélectionner la case à cocher **Activée** sur le panneau des interfaces Ethernet. Pour savoir comment accéder à ce panneau, voir Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

4. Cliquez sur **OK**.

   Avertissement : Une interface d'agrégation est activée par défaut. Pour continuer d'utiliser l'interface d'agrégation existante à la place de la nouvelle, vous devez d'abord désactiver cette dernière. Pour désactiver l'interface d'agrégation, développez Interfaces d'agrégation, puis recherchez la nouvelle interface d'agrégation que vous venez d'ajouter. Désélectionnez la case **Activée**. Si l'interface d'agrégation reste activée, lors du prochain redémarrage de WebSphere

DataPower XC10 Appliance, la nouvelle interface d'agrégation servira à associer les grilles de données, ce qui risque d'entraîner des résultats imprévus.

Remarque : Vous devez redémarrer pour que les modifications que vous apportez aux interfaces d'agrégation puissent être utilisées par WebSphere DataPower XC10 Appliance. Cependant, vous devez exécuter la commande `clear-all` puis la commande `device-restart` pour que ces modifications prennent effet.

Rubrique parent : Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

Tâches associées:
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Modification d'une interface d'agrégation

Vous pouvez ajouter ou supprimer des membres d'une interface d'agrégation et modifier des propriétés telles que l'adresse IP, la passerelle par défaut ou l'unité de transmission maximale (MTU). Vous pouvez également modifier la règle d'agrégation.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'interface de ligne de commande ou la console Web pour modifier une interface d'agrégation. Il est recommandé d'utiliser l'interface de ligne de commande. Avant chaque opération, vous devez vous connecter au dispositif. Après l'opération, exécutez la commande **`clear-all`** et redémarrez le dispositif.

Procédure

- Modification d'une interface d'agrégation à l'aide de l'interface de ligne de commande. Aidez-vous de l'exemple suivant pour modifier l'interface d'agrégation :

```
Console aggregate-interface:myAgg> aggregation-policy lacp
Console aggregate-interface:myAgg> transmit-hash-policy layer3+4
Console aggregate-interface:myAgg> exit
Console> show aggregate-interface myAgg
aggregate-interface myAgg: [Up]

name "myAgg"
AdminState "Enabled"
use-arp "true"
mtu "1500"
ip
  use-dhcp "false"
  address "10.5.5.5/24"
  use-slaac "false"
  dad-transmits "1"
  dad-retransmit-timer "1000"
end ip
aggregation-policy "lacp"
lacp-selection-logic "stable"
transmit-hash-policy "layer3+4"
member "eth3" "eth4"
primary-member "eth3"
```

- Modification d'une interface d'agrégation à l'aide de la console Web. Affichez le panneau Paramètres de l'une des manières suivantes :
  - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
  - Dans le panneau d'accueil, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.

  1. Développez l'entrée **Interfaces d'agrégation**.
  2. Vous pouvez ajouter ou supprimer des membres dans une interface d'agrégation existante. Une liste d'interfaces Ethernet membres de cette agrégation s'affiche.

     Conseil : Vous ne pouvez pas supprimer de membre s'il s'agit du seul membre, et vous ne pouvez pas supprimer un membre s'il est le membre principal.

     Important : Assurez-vous que l'interface Ethernet est désactivée avant de l'ajouter à l'interface d'agrégation. Pour plus d'informations sur l'activation ou la désactivation d'une interface Ethernet, voir Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance.

     - Pour ajouter d'autres membres, cliquez sur **Ajouter d'autres...**
     - Sélectionnez l'interface Ethernet, puis cliquez sur **OK**.
  3. Modifiez l'adresse IP et le masque de sous-réseau au format suivant : *<adresse_ip>/<masque_sous-réseau>*. Le masque de sous-réseau doit être indiqué en respectant la notation CIDR (Classless Inter-Domain Routing). Par exemple : *255.255.255.0* en notation longue devient *24* en notation CIDR.
  4. Modifiez la **passerelle par défaut**. Le dispositif utilise un routage basé sur la source et non sur la destination. Un paquet est envoyé vers sa destination via l'interface par laquelle il est arrivé.

Chaque interface dispose de sa propre table de routage, qui est distincte des autres interfaces. Pour chaque interface devant atteindre des destinations au-delà du sous-réseau local, indiquez une route par défaut accessible directement à partir de cette interface.

5. Modifiez l'**unité de transmission maximale (MTU)**. Cette zone spécifie la taille maximale, en octets, d'une unité de données du protocole lors d'une communication via une interface Ethernet. Sa valeur par défaut est de 1500 octets, ce qui est également la valeur maximale autorisée pour cette zone.
6. Cliquez sur **Editer** pour modifier la règle d'agrégation. En fonction de la règle d'agrégation à modifier, les propriétés ci-dessous s'affichent.

Règle d'agrégation

active-backup
Indique la règle active-backup pour la haute disponibilité. A l'aide de cette règle, une seule interface Ethernet (en tant que membre d'une interface d'agrégation) est activée à la fois. Si cette interface Ethernet échoue, un autre membre prend le relais. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre principal.
Membre principal
Indique l'interface Ethernet que vous souhaitez désigner comme membre principal. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre principal.

Règle d'agrégation

LACP
Indique la règle LACP (Link Aggregation Control Protocol) pour la haute disponibilité et la bande passante. Vous ne pouvez utiliser la règle LACP que lorsque son mode n'est pas paramétré sur 'OFF'. Par défaut, la logique de sélection est `Stable` et le mode de hachage de transmission est `layer2`.

Logique de sélection

- `Stable`

  Indique l'interface Ethernet ayant la bande passante la plus élevée. Lorsque la valeur stable est sélectionnée, l'interface Ethernet est de nouveau sélectionnée lorsqu'une interface d'agrégation activée ne possède aucun membre disponible. Stable est la valeur par défaut.

- `Bande passante`

  Indique l'interface Ethernet ayant la bande passante la plus élevée. Cette interface Ethernet est de nouveau sélectionnée lorsqu'un autre membre est ajouté ou supprimé ou que l'interface d'agrégation est activée ou désactivée.

- `Nombre`

  Indique l'interface d'agrégation ayant le nombre le plus élevé d'interfaces Ethernet comme membres.

Règle de hachage

- `layer2`

  Indique le résultat OU exclusif (XOR) des adresses MAC afin de générer un hachage.

- `layer2+3`

  Indique le résultat XOR des adresses MAC et des adresses IP afin de générer un hachage.

- `layer3+4`

  Indique le résultat XOR des adresses IP et des numéros de port afin de générer un hachage.

Règle d'agrégation

balance-tlb
Indique la règle balance-tlb pour l'équilibrage de charge et la haute disponibilité. Cette règle distribue le trafic sortant en fonction de la charge en cours de chaque membre. Le trafic entrant est systématiquement transmis à l'interface Ethernet sélectionnée comme membre principal. Si l'interface Ethernet de réception échoue, un autre membre prend le relais.
Membre principal
Indique l'interface Ethernet que vous souhaitez désigner comme membre principal. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre

principal.

7. Cliquez sur **OK**.

   Important : Lorsque vous modifiez une interface d'agrégation, les modifications ne prennent effet que lorsque vous redémarrez le dispositif. Ce redémarrage est nécessaire pour que les processus WebSphere DataPower XC10 Appliance se lient à l'interface Ethernet. Vous devez désactiver l'interface d'agrégation modifiée avant de redémarrer WebSphere DataPower XC10 Appliance. Pour désactiver l'interface, développez **Interfaces d'agrégation**, puis recherchez l'interface d'agrégation que vous venez de modifier. Désélectionnez la case **Activée**. Après avoir désactivé l'interface d'agrégation, vous devez exécuter les commandes `clear-all` et `device-restart` dans l'interface de ligne de commande. Si vous n'exécutez pas ces commandes, les modifications ne prennent pas effet.

Rubrique parent : Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

Tâches associées:
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Suppression d'une interface d'agrégation

Vous pouvez supprimer une interface d'agrégation.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'interface de ligne de commande ou la console Web pour supprimer une interface d'agrégation. Il est recommandé d'utiliser l'interface de ligne de commande. Avant chaque opération, vous devez vous connecter au dispositif. Après l'opération, exécutez la commande **`clear-all`** et redémarrez le dispositif.

Procédure

- Suppression d'une interface d'agrégation à l'aide de l'interface de ligne de commande. Aidez-vous de l'exemple suivant pour supprimer l'interface d'agrégation :

```
Console> delete aggregate-interface myAgg
Console> list aggregate-interface
```

- Suppression d'une interface d'agrégation à l'aide de la console Web. Affichez le panneau Paramètres de l'une des manières suivantes :
  - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
  - Dans le panneau d'accueil, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.

  1. Cliquez sur l'icône de suppression ( ) pour supprimer l'interface d'agrégation.

     Important : Lorsque vous supprimez une interface d'agrégation, les modifications ne prennent effet que lorsque vous redémarrez le dispositif. Après avoir supprimé l'interface d'agrégation, vous devez exécuter les commandes `clear-all` et `device-restart` dans l'interface de ligne de commande. Si vous n'exécutez pas ces commandes, WebSphere DataPower XC10 Appliance continue à utiliser cette interface.

Rubrique parent : Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

Tâches associées:
Modification d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Importation et exportation de configurations

Lorsque vous configurez un nouveau dispositif, vous pouvez exporter et stocker les paramètres de configuration pour ce dispositif. Si ultérieurement des modifications sont effectuées qui nécessitent la suppression ou la réinstallation du dispositif, vous pouvez alors importer les données de configuration stockées sans perdre les paramètres de configuration.

Avant de commencer

Pour importer ou exporter des configurations, vous devez tout d'abord configurer l'environnement de votre dispositif avec des grilles de données, des utilisateurs ou des groupes, ainsi que des informations LDAP. Pour plus d'informations, voir Configuration de votre dispositif.

Pourquoi et quand exécuter cette tâche

En tant qu'administrateur, vous pouvez configurer des dispositifs, ce qui implique de créer des utilisateurs et des groupes pour la grille de données ainsi que des paramètres de configuration LDAP. Une fois cette configuration effectuée, vous pouvez conserver ces informations en exportant ces configurations dans un fichier. Ensuite, vous pouvez utiliser ce fichier exporté ultérieurement pour réimporter dans le dispositif les informations de configuration exportées. Vous pouvez importer et exporter les informations de configuration suivantes :

- Configurations de grille de données
- Configurations utilisateur
- Configuration de groupe
- Configurations LDAP

Vous devrez peut-être importer et exporter les données de configuration si des erreurs se produisent qui nécessitent que vous supprimiez un dispositif d'une collectivité ou que vous réinstalliez un dispositif dans un environnement autonome. Par exemple, lorsque vous exportez des informations de configuration depuis un dispositif que vous ajoutez à une collectivité, vous pouvez utiliser la fonction d'importation pour récupérer la configuration du dispositif avant que ce dernier rejoigne la collectivité. Dans les environnements autonomes ou de collectivité, l'utilisation des actions d'importation et d'exportation vous permettent de gagner du temps puisque vous n'avez pas à recréer manuellement les informations de configuration pour le dispositif supprimé.

Pour les environnements autonomes : Si vous utilisez un dispositif autonome, effectuez les étapes 1 et 4, qui incluent la syntaxe de commande **`config`** suivante pour l'importation et l'exportation de configurations :

```
config <import|export> -file <nom_fichier> [-silent]
```

La commande **`config export`** accepte également les paramètres dont la valeur est 0. Dans ce cas, la commande effectue une exportation archivée, que vous ne pouvez pas spécifier à l'aide de l'indicateur `-file`.

Pour plus d'informations sur l'utilisation des commandes, entrez `config usage`.

Procédure

1. Exporter vos informations de configuration dans un fichier que vous pouvez utiliser ultérieurement. Par exemple, effectuez une exportation dans le fichier `foo.json`. Utilisez l'interface de ligne de commande, puis exécutez la commande **`config`**, suivie d'une liste des paramètres possibles, par exemple :

   ```
   config <export> -file foo.json -silent
   ```

   Dans l'exemple précédent, le paramètre **`-silent`** est facultatif. Si vous précisez l'indicateur **`-silent`**, les messages d'état de configuration ne s'affichent pas à l'écran (ces messages sont affichés par défaut).
2. Facultatif : Ajouter le nouveau dispositif à une collectivité.
3. Supprimer le dispositif de la collectivité lorsqu'une erreur se produit qui endommage vos données de configuration. Vous pouvez utiliser les outils d'analyse de journal pour analyser le fonctionnement de l'environnement d'exécution et résoudre les problèmes qui se produisent dans l'environnement de votre dispositif.
4. Importer le fichier `foo.json` créé lors de la première étape. Utilisez l'interface de ligne de commande, puis exécutez la commande **`config`**, suivie d'une liste des paramètres possibles, par exemple :

   ```
   config <import> -file foo.json -silent
   ```

Dans l'exemple précédent, le paramètre **-silent** est facultatif. Si vous précisez l'indicateur **-silent**, les messages d'état de configuration ne s'affichent pas à l'écran, ces messages étant affichés par défaut.

5. Créer une nouvelle collectivité avec ce dispositif comme principal.
6. Ajouter d'autres dispositifs à la nouvelle collectivité, un par un.

2.5+ **Exportation de configurations**

Vous pouvez spécifier quand les configurations doivent être exportées à partir de WebSphere DataPower XC10 Appliance.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Exportation de configurations

Vous pouvez spécifier quand les configurations doivent être exportées à partir de WebSphere DataPower XC10 Appliance.

Pourquoi et quand exécuter cette tâche

Les configurations peuvent être exportées selon une périodicité mensuelle, hebdomadaire ou quotidienne, et vous pouvez même préciser l'heure à laquelle elles doivent avoir lieu. Par exemple, vous pouvez spécifier qu'une configuration de grille de données doit être exportée le premier jour du mois, et que les configurations de ce type doivent être conservées dans l'historique pendant une durée d'un an.

Le nombre maximum de configurations exportées pouvant être stockées est de 10. Ce total inclut les exportations créées manuellement et celles créées à l'aide de la console Web.

Procédure

1. Dans la console Web, cliquez sur **Dispositif** > **Importation et exportation de configuration**.
2. Développez la section Export.
    a. Sélectionnez **Planifier les exportations**.
    b. Spécifiez la date et l'heure de début de l'exportation.
    c. Spécifiez la durée de validité de ce planning.
3. Cliquez sur le **bouton de mise à jour du planning**.
4. Facultatif : Si vous voulez exporter une configuration entre deux exportations planifiées, vous devez l'exporter manuellement. Pour ce faire, cliquez sur le **bouton d'exportation immédiate**.

Rubrique parent : Importation et exportation de configurations

Gestion du serveur DNS (Domain Name System)

Un serveur DNS (Domain Name System) est requis pour le système IBM® WebSphere DataPower XC10 Appliance étant donné que les services de recherche DNS sont utilisés pour la communication. Vous devez spécifier ce serveur DNS lors de l'initialisation du dispositif.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Un serveur DNS est requis pour que le dispositif fonctionne correctement. Vos serveurs DNS doivent comporter des entrées DNS directes et inverses pour la plage d'adresses DNS gérées par WebSphere DataPower XC10 Appliance. WebSphere DataPower XC10 Appliance utilise le nom d'hôte dérivé de la recherche inversée lors du déploiement d'un système virtuel. Si cette recherche échoue du fait qu'aucun nom d'hôte n'a été défini, le déploiement ne peut pas aboutir, car il nécessite un nom d'hôte et pas seulement une adresse

Procédure

1. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Serveurs de noms de domaine**.
3. Facultatif : Cliquez sur son **<adresse IP>** pour modifier un serveur DNS existant.
4. Sélectionnez **Cliquez pour ajouter** afin d'ajouter un nouveau serveur DNS. Un serveur DNS est configuré lors de l'initialisation du dispositif.
5. Facultatif : Cliquez sur l'icône de suppression ( ) pour supprimer un serveur DNS.

Résultats

A l'aboutissement de ces étapes, vous aurez défini un serveur DNS à utiliser pour les recherches lors des communications.

Que faire ensuite

Vous pouvez également définir des serveurs DNS avec l'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.

Rubrique parent : Configuration de votre dispositif

Concepts associés:
Mot de passe xcadmin

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Mappage d'adresses IP vers des noms d'hôte

Avant que les informations relatives à une adresse puissent être utilisées pour créer une connexion dans un protocole de réseau TCP/IP, l'adresse IP doit être associée à un nom d'hôte. Vous pouvez résoudre une adresse IP en nom d'hôte en éditant le fichier etc/hosts du dispositif. Vous pouvez éditer le fichier etc/hosts à partir de l'interface utilisateur.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Editez le fichier etc/hosts à partir de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez **Adresses IP vers noms d'hôte**.
3. Cliquez sur **Créer un mappage** pour spécifier l'adresse IP et le nom d'hôte.
4. Cliquez sur **OK** pour créer le mappage. Ce mappage édite le fichier etc/hosts du dispositif.
5. Facultatif : Pour supprimer un mappage, cliquez sur l'icône de suppression ( ).

Résultats

A l'issue de ces étapes, vous aurez édité le fichier etc/hosts qui associe une adresse IP à un nom d'hôte.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Gestion des paramètres de date et heure

Utilisez des serveurs NTP (Network Time Protocol) pour maintenir la synchronisation de la date et de l'heure entre les différents dispositifs de votre collectivité.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Il est important de configurer un serveur NTP lorsque vous utilisez une collectivité. Ce serveur permet d'établir une corrélation entre les journaux des différents dispositifs de la collectivité et donc de garantir la cohérence des dates et des heures au niveau des entrées de journal.

Procédure

1. Accédez au panneau Paramètres. Pour gérer vos paramètres de date et d'heure, accédez au panneau Paramètres à l'aide d'une des méthodes suivantes :
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Date et heure**.
3. Sélectionnez la valeur appropriée dans le menu déroulant **Le fuseau horaire actuel est**.
4. Sélectionnez sur **Cliquez pour ajouter** afin d'ajouter un nouveau serveur NTP. Par défaut, aucun serveur NTP n'est configuré.
5. Cliquez sur le nom du serveur et faites-le glisser pour réorganiser vos serveurs NTP. Le premier serveur NTP disponible dans la liste sera utilisé pour maintenir la synchronisation.
6. Cliquez sur l'icône de suppression ( ) pour supprimer un serveur NTP.
7. Redémarrez le dispositif pour que vos modifications prennent effet. Pour plus d'informations, voir Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Résultats

Vous avez défini un serveur NTP pour maintenir les horloges synchronisées dans les dispositifs de la collectivité.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance

La fonction de distribution du courrier du dispositif est utilisée afin de redéfinir les mots de passe des utilisateurs. Lorsqu'un utiliseur demande un nouveau mot de passe, il le reçoit dans un courrier électronique généré par le dispositif.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Vous devez configurer un serveur SMTP (Simple Mail Transfer Protocol) pour son utilisation avec WebSphere DataPower XC10 Appliance. La fonction de distribution du courrier est utilisée pour envoyer un nouveau mot de passe à l'utilisateur en cas d'oubli. Si l'utilisateur a oublié son mot de passe alors qu'un serveur SMTP n'a pas été configuré, l'utilisateur ne peut pas recevoir un nouveau mot de passe.

Procédure

1. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Distribution du courrier**.
3. Ajoutez un serveur SMTP. Indiquez l'adresse IP ou le nom d'hôte du serveur SMTP à utiliser pour WebSphere DataPower XC10 Appliance. Si un nom d'hôte est utilisé pour cette zone, il doit pouvoir être résolu par les serveurs DNS (Domain Name System) définis pour le dispositif. Pour plus d'informations sur l'ajout d'un serveur DNS, reportez-vous à la rubrique Gestion du serveur DNS (Domain Name System).
4. Ajoutez une adresse de réponse. Utilisez pour cette zone l'adresse électronique de l'administrateur.

Résultats

Vous avez spécifié un serveur SMTP et une adresse de réponse à utiliser pour la réinitialisation des mots de passe. Un lien **Mot de passe oublié ?** s'affiche dans l'écran de connexion pour que les utilisateurs puissent réinitialiser leurs mots de passe.

Rubrique parent : Configuration de votre dispositif

Concepts associés:
Mot de passe xcadmin

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Le système WebSphere DataPower XC10 Appliance peut être redémarré ou arrêté à partir de l'interface utilisateur.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces opérations. Lorsque vous arrêtez ou que vous redémarrez le dispositif, vous devez dans un premier temps vérifier que tous les processus en cours d'exécution sont terminés. Pour contrôler la progression des processus en cours d'exécution, cliquez sur **Tâches**.

Pourquoi et quand exécuter cette tâche

Vous pouvez arrêter ou redémarrer WebSphere DataPower XC10 Appliance à partir de l'interface utilisateur.

Procédure

1. Accédez au panneau **Paramètres**. Pour gérer la mise sous tension de votre dispositif, accédez au panneau **Paramètres** à l'aide d'une des méthodes suivantes :
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Identification et résolution des incidents**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Alimentation**.
3. Cliquez sur **Redémarrer le dispositif** pour le redémarrer. Pendant le redémarrage, tous les logiciels du dispositif sont arrêtés, puis le dispositif est redémarré. Vous pouvez choisir de redémarrer le dispositif immédiatement ou attendre que toutes les tâches actives soient terminées avant de redémarrer le dispositif.
4. Cliquez sur **Arrêter le dispositif** pour arrêter celui-ci. Pendant l'arrêt, tous les logiciels du dispositif sont arrêtés, puis le dispositif est arrêté. Vous pouvez choisir d'arrêter le dispositif immédiatement ou attendre que toutes les tâches actives soient terminées avant de l'arrêter. Pour interrompre l'alimentation du dispositif, vous devez actionner l'interrupteur d'alimentation physique placé au dos du dispositif afin d'éteindre celui-ci.

Résultats

2.5+ Vous pouvez suivre l'état de démarrage du dispositif dans l'interface utilisateur. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.

A l'issue de cette procédure, l'appareil sera arrêté ou redémarré, en fonction de votre sélection.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Configuration des collectivités et des zones

Les collectivités et les zones permettent de définir la topologie physique (emplacements) de vos dispositifs WebSphere DataPower XC10 Appliance.

1. Création d'une collectivité
   Pour regrouper plusieurs WebSphere DataPower XC10 Appliance à des fins de disponibilité, d'évolutivité et de gestion, vous devez créer une collectivité.

2. Création et modification des zones
   Les zones permettent d'indiquer l'emplacement physique d'un dispositif. Cet emplacement physique peut correspondre à deux emplacements différents dans le même centre de données ou à des dispositifs situés dans des centre de données différents, éventuellement dans différentes régions du pays.

3. Configuration d'une réplication multimaître entre les collectivités
   La réplication multimaître est une technique qui permet de garantir une disponibilité continue pour plusieurs environnements de déploiement. En établissant des liaisons de collectivité entre les serveurs de catalogue dans vos collectivités, vous pouvez répliquer des données de manière asynchrone entre les grilles de données de deux collectivités différentes.

4. 2.5+ Configuration d'IBM eXtremeIO (XIO)
   IBM® eXtremeIO (XIO) est un mécanisme de transport qui remplace ORB (Object Request Broker).

Création d'une collectivité

Pour regrouper plusieurs WebSphere DataPower XC10 Appliance à des fins de disponibilité, d'évolutivité et de gestion, vous devez créer une collectivité.

Avant de commencer

Avant de définir une collectivité, vous devez posséder les droits d'accès d'administration du dispositif sur les consoles des dispositifs que vous souhaitez ajouter à la collectivité.

Pourquoi et quand exécuter cette tâche

Pour créer une collectivité, vous devez ajouter tous les noms d'hôte et clés secrète de tous les dispositifs à regrouper dans une collectivité à la configuration de l'un des dispositifs. Les dispositifs peuvent communiquer toutes les informations de configuration. Ces informations incluent les grilles de données, les informations de contrôle, les utilisateurs et les groupes et les appartenances à la collectivité.

Tous vos dispositifs doivent être au même niveau de microprogramme avant que vous puissiez les ajouter à une collectivité. Pour plus d'informations sur l'exécution des mises à niveau de microprogramme, voir Mise à jour du microprogramme.

Procédure

1. Sélectionnez un dispositif auquel vous allez ajouter les noms d'hôtes et les clés secrètes des autres dispositifs de la collectivité. Pour tous les autres dispositifs, ouvrez une nouvelle fenêtre de navigateur et effectuez la procédure suivante :
   a. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres**.
   b. La liste des dispositifs de la collectivité s'affiche. Le dispositif auquel vous êtes connecté est signalé avec l'icône de console ( ). Notez le **nom d'hôte** et **la clé secrète** de ce dispositif.

   Tableau 1. Etats de dispositif

Icône	Définition
	Indique le dispositif auquel vous êtes connecté et que le dispositif est actif.
	Indique que le dispositif est démarré et qu'il peut être configuré et utilisé.
	Indique que le dispositif est arrêté et qu'il n'est pas disponible.

2. Ajoutez tous les noms d'hôtes et les clés secrètes de chaque dispositif à l'un des dispositifs du groupe.
   a. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres**.
   b. Cliquez sur l'icône Ajouter ( ).
   c. Accédez à **Nom d'hôte** et **Clé secrète** pour le dispositif à ajouter à la collectivité.
3. Contrôlez la progression de l'ajout des dispositifs à la collectivité dans le vue Tâches. Dans l'interface utilisateur, cliquez sur **Tâches**.

Résultats

Les dispositifs sont regroupés dans une collectivité afin de faciliter l'évolutivité et la gestion.

Rubrique parent : Configuration des collectivités et des zones
Rubrique suivante : Création et modification des zones

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Création et modification des zones

Les zones permettent d'indiquer l'emplacement physique d'un dispositif. Cet emplacement physique peut correspondre à deux emplacements différents dans le même centre de données ou à des dispositifs situés dans des centre de données différents, éventuellement dans différentes régions du pays.

Avant de commencer

Avant de créer et de configurer des zones, vous devez installer, configurer et assembler vos dispositifs au sein d'une collectivité. Vous pouvez alors sélectionner les dispositifs de votre collectivité et les répartir dans des zones spécifiques. Si vous ajoutez un dispositif à la collectivité après la création des zones, vous pouvez spécifier la zone lors de l'ajout du dispositif.

Pourquoi et quand exécuter cette tâche

En spécifiant des zones de manière à indiquer l'emplacement physique des dispositifs, les données stockées dans les grilles de données peuvent être placées à l'emplacement approprié. DataPower XC10 Appliance utilise des zones pour déterminer l'emplacement des informations principales et secondaires du cache. Par exemple, si une grille de données principale se trouve dans la zone 1, la réplique de cette grille de données se trouve en zone 2. Si vous n'avez qu'une seule zone, aucune réplique n'est créée.

Lors de la première initialisation d'un dispositif, le dispositif est ajouté à la zone par défaut appelée **DefaultZone**. Lorsque vous ajoutez les dispositifs aux zones que vous créez, ils sont supprimés de la zone **DefaultZone**. Vous ne pouvez pas remplacer les dispositifs dans la zone **DefaultZone** après les en avoir retirés.

Procédure

1. Dans l'interface utilisateur, cliquez sur**Collectivité** > **Zones**.
2. Ajoutez des zones. Cliquez sur **Ajouter des zones**. Indiquez le nom de la zone à ajouter. Le nom de la zone s'affiche sur la page de la console. Vous pouvez répéter cette procédure si vous devez créer plusieurs zones.
3. Ajoutez des dispositifs à la zone. Développez le nom de la zone et sélectionnez les dispositifs à ajouter à la zone. Répétez cette procédure jusqu'à ce que tous les dispositifs de la collectivité aient été ajoutés aux zones. Chaque dispositif doit se trouver dans une zone. Une même zone ne peut pas contenir de dispositifs de collectivités différentes.
4. Appliquez les modifications de zone. Les modifications apportées aux zones sont effectives lorsque vous cliquez sur **Activer les modifications de zone**. Pour modifier les informations de zone, vous pouvez cliquer sur l'icône de suppression ( ) pour supprimer une zone ou cliquer sur **Réinitialiser les modifications de zone** pour restaurer la dernière configuration de zone activée.

Résultats

DataPower XC10 Appliance peut créer des zones de cache principales et secondaires en fonctions des emplacements physiques (zones) définis. Vous pouvez afficher ou modifier la topologie des zones à tout moment en cliquant sur **Collectivité** > **Zones** dans l'interface utilisateur.

Que faire ensuite

Créez et configurez vos applications et vos grilles de données.

Rubrique parent : Configuration des collectivités et des zones
Rubrique précédente : Création d'une collectivité
Rubrique suivante : Configuration d'une réplication multimaître entre les collectivités

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Configuration d'une réplication multimaître entre les collectivités

La réplication multimaître est une technique qui permet de garantir une disponibilité continue pour plusieurs environnements de déploiement. En établissant des liaisons de collectivité entre les serveurs de catalogue dans vos collectivités, vous pouvez répliquer des données de manière asynchrone entre les grilles de données de deux collectivités différentes.

Avant de commencer

- Au moins deux collectivités doivent être configurées. Pour que les données soient répliquées entre les collectivités, vous devez créer des configurations de grille de données identiques dans chaque collectivité.
- Déterminez le type de topologie de collectivité multiple que vous souhaitez définir. Pour plus d'informations sur les topologies multimaîtres, voir Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître.
- Vous pouvez configurer des liaisons entre les collectivités dans l'interface utilisateur ou à l'aide de l'utilitaire **xscmd**.

Pourquoi et quand exécuter cette tâche

Une seule collectivité ne couvre pas un réseau non fiable car des incidents de faux positif risquent d'être détectés. Cependant, vous pouvez être amené à répliquer des données de grille de données sur des dispositifs dont la connectivité réseau n'est pas fiable. Voici quelques scénarios courants dans lesquels vous pouvez être amené à utiliser ce type de topologie :

- Reprise après incident entre des centres de données dans lesquels une collectivité est active et une autre est utilisée aux fins de secours.
- Centres de données géographiquement répartis dans lesquels toutes les collectivités sont actives pour les clients géographiquement proches.

Une fois que vous connectez deux collectivités, toutes les grilles de données portant le même nom sont répliquées de façon asynchrone d'une collectivité à l'autre. Ces grilles de données doivent comporter le même nombre de répliques dans chaque collectivité et posséder les mêmes configurations de mappe dynamique.

Procédure

1. Etablissez une liaison entre les collectivités.
   - **Etablissement de liaisons de collectivité à l'aide de l'interface utilisateur :**
     a. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Liaisons de collectivité**.
     b. Cliquez sur l'icône d'ajout () et entrez le nom d'hôte ou l'adresse IP, le nom d'utilisateur et le mot de passe du dispositif distant.
   - **Etablissement de liaisons de collectivité à l'aide de l'utilitaire `xscmd` :**
     a. Extrayez l'adresse IP et le port des serveurs de catalogue dans chaque collectivité. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres** > ***nom_membre_collectivité***. Une liste de serveurs de catalogue et de numéros de port s'affiche. Répétez cette étape pour chaque collectivité à connecter.
     b. Connectez l'utilitaire **xscmd** à l'une des collectivités. Pour plus d'informations sur le démarrage de l'utilitaire **xscmd**, voir Administration avec l'utilitaire xscmd.

        ```
        xscmd.sh -ts xsatruststore.jks -tst jks -ssl -tsp xc10pass
        -user xcadmin -tsp xcadmin -cep myxc10.mycompany.com:2809
        [additional_xscmd_parameters]
        ```

     c. A partir de la collectivité à laquelle vous êtes connecté, exécutez la commande suivante :

        ```
        xscmd –c establishLink -cep myxc10.mycompany.com:2809 -fd dname
        -fe myxc102.mycompany.com:2809,myxc103.mycompany.com:2809
        ```

        Le paramètre **-fd dname** spécifie le nom de la collectivité éloignée avec laquelle vous voulez établir une liaison. Dans l'exemple ci-dessus, le nom de la collectivité éloignée est **dname**. Pour trouver le nom d'une collectivité, consultez la liste des serveurs de catalogue sur le panneau des informations détaillées sur les membres. Le nom de la collectivité est la première adresse IP de la liste.

Une fois la liaison établie, les serveurs de catalogue des collectivités commencent à se répliquer mutuellement. Il n'est pas nécessaire d'établir la liaison dans les deux sens.

2. Vous pouvez également annuler des liaisons entre les collectivités.
    - **Annulation de liaisons de collectivité à l'aide de l'interface utilisateur :** Dans l'interface utilisateur, cliquez sur l'icône de suppression () pour supprimer une liaison sélectionnée.
    - **Annulation de liaisons de collectivité à l'aide de l'utilitaire `xscmd` :** A l'aide de l'utilitaire **`xscmd`**, exécutez la commande suivante :

      ```
      xscmd –c dismissLink -cep myxc10.mycompany.com:2809 -fd dname
      ```

Rubrique parent : Configuration des collectivités et des zones
Rubrique précédente : Création et modification des zones
Rubrique suivante : 2.5+ Configuration d'IBM eXtremeIO (XIO)

Concepts associés:
Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître
Topologie du dispositif : collectivités, zones et grilles de données

Affichage du type de transport du domaine de service de catalogue

Vous pouvez afficher le type de transport qui est actuellement utilisé pour le domaine de service de catalogue.

Avant de commencer

Vous pouvez afficher les types de transports qui sont utilisés dans un domaine de service de catalogue autonome ou un domaine de service de catalogue en cours d'exécution dans WebSphere Application Server.

- Si vous utilisez un domaine de service de catalogue autonome, utilisez l'utilitaire **xscmd** pour afficher les informations de transport sur le domaine de service de catalogue. Pour plus d'informations sur la configuration de l'utilitaire **xscmd**, voir Administration avec l'utilitaire xscmd.
- Si vous disposez d'un domaine de service de catalogue qui s'exécute dans WebSphere Application Server, vous pouvez utiliser l'utilitaire **wsadmin** pour afficher le type de transport. Pour plus d'informations sur l'utilitaire **wsadmin**, voir Démarrage du client de scriptage wsadmin en utilisant le scriptage wsadmin.

Procédure

- Affichez le type de transport d'un domaine de service de catalogue autonome. Dans l'utilitaire **xscmd**, exécutez la commande suivante :
  - UNIX `./xscmd.sh -c showTransport`
  - Windows `xscmd.bat -c showTransport`

  La commande affiche le type de transport. Les valeurs suivantes peuvent s'afficher : `eXtremeIO` ou `Object Request Broker`.
- Affichez le type de transport d'un domaine de service de catalogue qui s'exécute WebSphere Application Server.
  - Dans la console d'administration, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaine de service de catalogue** > ***nom_domaine_service_catalogue***. Vérifiez que **Activer IBM eXtremeIO (XIO) communication** est sélectionné.
  - Dans l'utilitaire **wsadmin**, exécutez la commande suivante :
    - A l'aide de Jacl :

      ```
      $AdminTask getTransport {-domainName TestDomain }
      ```

    - A l'aide de la chaîne Jython :

      ```
      AdminTask.getTransport ('[-domainName testDomain]')
      ```

    La commande affiche le type de transport. Les valeurs suivantes peuvent s'afficher : `eXtremeIO` ou `Object Request Broker`. Si vous exécutez cette commande sur un domaine de service de catalogue qui contient des serveurs distants ou que catalogServerName est un serveur distant, une erreur se produit. Vous devez utiliser la commande **xscmd -c showTransport** pour les serveurs distants. Pour plus d'informations sur la commande **getTransport** dans l'utilitaire **wsadmin**, voir Tâches d'administration des domaines de service de catalogue.

Rubrique parent : 2.5+ Configuration d'IBM eXtremeIO (XIO)

Configuration des grilles de données

Vous pouvez créer trois types différents de grilles de données : des grilles de données simples grilles de données, de session grilles de données et de mémoire cache dynamique.

Création de grilles de données simples
Les grille de données simples permettent d'effectuer des opérations de création, de récupération, de mise à jour et de suppression. Une grille de données simple permet notamment d'accélérer l'accès aux données sur une base de données.

Création de grilles de données de session
Vous pouvez créer des grilles de données pour enregistrer des sessions HTTP à partir de vos applications Java ou .NET.

Java **Création de grilles de données en cache dynamique**
IBM® WebSphere DataPower XC10 Appliance permet de stocker des données pour une instance de cache dynamique WebSphere Application Server. En configurant cette fonction, vous permettez aux applications qui utilisent une instance de cache dynamique WebSphere Application Server d'utiliser les fonctions et les fonctionnalités de performance du dispositif.

Configuration de la capacité maximale d'une grille de données
Vous pouvez définir une capacité maximale pour chaque grille de données de la collectivité. La configuration d'une capacité maximale limite la quantité de stockage de données qui peut être utilisée par une grille de données. La limite de capacité permet de s'assurer que la capacité de stockage disponible pour la collectivité est utilisée de façon prévisible.

Activation de la sécurité pour les grilles de données
Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

Effacement des grilles de données
Vous pouvez supprimer définitivement toutes les entrées d'une grille de données. Vous pouvez effacer la grille de données pour supprimer les informations périmées ou tester les entrées.

Suppression des grilles de données
Si vous souhaitez supprimer les données d'une grille de données, vous pouvez supprimer la grille de données puis la recréer.

Configuration d'un fournisseur de cache Spring
Spring Framework Version 3.1 a introduit une nouvelle abstraction de cache. Celle-ci vous permet d'ajouter de manière transparente la mise en cache à une application Spring existante. Vous pouvez utiliser WebSphere DataPower XC10 Appliance comme fournisseur de cache pour l'abstraction de cache.

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Création de grilles de données simples

Les grille de données simples permettent d'effectuer des opérations de création, de récupération, de mise à jour et de suppression. Une grille de données simple permet notamment d'accélérer l'accès aux données sur une base de données.

Avant de commencer

Les grilles de données simples peuvent s'utiliser avec WebSphere Application Server ou avec une application Java™ autonome. Le client WebSphere eXtreme Scale Client doit être installé dans tous les cas.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser les grilles de données simples pour accélérer les applications Web dynamiques Web en allégeant la charge de la base de données. Vous pouvez stocker les paires clé-valeur de données arbitraires en mémoire, réduisant ainsi les requêtes de base de données lourdes. Les clés peuvent correspondre à n'importe quel type Java java.lang.String ou Entier. Les valeurs peuvent correspondre à n'importe quel type d'objet. A chaque fois que des données doivent être utilisées, la grille de données simple du dispositif est consultée en premier. Si les données ne se trouvent pas sur le dispositif, elles sont récupérées à partir de la base de données, puis insérées dans la grille de données.

Procédure

1. Créez la grille de données simple. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Grille de données simple**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données simple grille de données à créer. Les caractères suivants ne peuvent pas être utilisés dans le nom de la grille de données : `^ . \\ / , # $ @ : ; \ * ? < > | = + & % [ ] " "`.
2. Téléchargez le fichier `objectgrid.xml` pour votre grille de données simple. Dans la configuration de la grille de données simple que vous avez créée, cliquez sur l'icône de téléchargement ( ) et enregistrez le fichier sur votre disque local.
3. Créer une application qui accède à la grille de données. Pour plus d'informations, voir Développement d'applications pour accéder à des grilles de données simples.

Que faire ensuite

- Configurez la sécurité avant de commencer à envoyer des données à la grille de données. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
- Configurez des répliques. Les répliques permettent de s'assurer que les données de vos grilles de données sont disponibles si la copie principale échoue. Pour configurer des répliques, cliquez sur **Grille de données** > **Grille de données simple** > **Afficher les attributs avancés**. Les répliques ne sont créées que si le dispositif est dans une collectivité. Si le nombre de dispositifs dans la collectivité est *n*, le nombre maximal de répliques est *n-1*. Ainsi, si vous configurez trois répliques mais que vous n'avez que deux dispositifs dans la collectivité, une seule réplique est créée. Des répliques supplémentaires sont créées si vous ajoutez des dispositifs à la collectivité. Définissez le nombre de réplique au nombre idéal de répliques souhaitées : ainsi, lorsque des dispositifs rejoignent la collectivité, de nouvelles répliques sont créées. Le contenu de la grille de données est effacé lorsque vous modifiez le nombre de répliques.
- Configurez une limite de capacité pour la grille de données. En configurant des limites de capacité sur la grille de données, vous vous assurez que la capacité de stockage pour la collectivité est utilisée de façon prévisible. Pour plus d'informations, voir Configuration de la capacité maximale d'une grille de données.
- Configurez un expulseur TTL pour une grille simple. Pour plus d'informations, voir Configuration d'un expulseur TTL (Time To Live).
- Vous pouvez contrôler votre grille de données dans l'interface utilisateur de DataPower XC10 Appliance. Pour plus d'informations, voir Surveillance des grilles de données dans l'interface utilisateur.

Configuration des mappes dynamiques
Vous pouvez créer de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Configuration d'une stratégie de verrouillage
Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale. Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste.

Configuration d'un expulseur TTL (Time To Live)

Lorsque vous créez une grille simple, une mappe par défaut (statique) et un ensemble de mappes dynamiques sont créés. Par défaut, aucun expulseur TTL n'est configuré pour une mappe par défaut. Si vous disposez d'une mappe dynamique, vous pouvez définir une valeur de durée de vie (TTL) pour la date/heure de création (.*CT), la date/heure de la dernière mise à jour (*.LUT) ou la date/heure du dernier accès (*.LAT). Vous pouvez modifier ce comportement par défaut de sorte qu'un expulseur TTL soit également activé pour une mappe par défaut.

Java **Configuration du cache local**

Vous pouvez configurer vos clients pour un cache local en ligne L'on appelle cache local ce cache facultatif. Il s'agit d'une grille de données indépendante, présente sur chaque client et faisant office de cache du cache distant côté serveur. Le cache local est activé par défaut lorsque le verrouillage est désactivé ou configuré comme optimiste et ne peut pas être utilisé lorsqu'il est défini comme pessimiste.

2.5+ **Gestion des noms d'hôte pour la communication entre les clients et les serveurs**

Utilisez la propriété **`publishHost`** pour configurer le nom d’hôte que le transport par réseau publie à destination des serveurs et clients connectés.

Rubrique parent : Configuration des grilles de données

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Configuration des mappes dynamiques

Vous pouvez créer de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Avant de commencer

- Créez une grille de données simple. Pour plus d'informations, voir Création de grilles de données simples.
- Choisissez les options de configuration que vous souhaitez utiliser dans votre mappe dynamique. Pour plus d'informations, voir Options de configuration de mappe dynamique.

Que faire ensuite

Créez une mappe dynamique à partir de vos modèles définis :

- Java **Avec des API Java** Voir Création de mappes dynamiques avec les API Java pour un exemple d'appel de la Session.getMap(String) pour définir votre mappe dynamique.
- .NET 2.5+ **Avec des API .NET** Voir Création de mappes dynamiques avec les API .NET pour plus d'informations.
- **Avec la passerelle REST :** Voir Exemple de passerelle REST : Création de mappes dynamiques pour plus d'informations.

Rubrique parent : Création de grilles de données simples

Tâches associées:
Configuration d'une stratégie de verrouillage
Configuration d'un expulseur TTL (Time To Live)
Java Configuration du cache local
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs
Java 2.5+ Configuration de l'invalidation du cache local

Référence associée:
Options de configuration de mappe dynamique

Configuration d'une stratégie de verrouillage

Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale. Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste.

Avant de commencer

Déterminez la stratégie de verrouillage à utiliser. Pour plus de détails, voir Stratégies de verrouillage.

Pourquoi et quand exécuter cette tâche

Chaque instance de mappe de sauvegarde peut être configurée pour utiliser l'une de ces stratégies de verrouillage :

- Mode de verrouillage optimiste (valeur par défaut)
- Mode de verrouillage pessimiste (requis pour les applications .NET)
- Aucun

Procédure

Pour savoir comment configurer un mode de verrouillage optimiste, un mode de verrouillage pessimiste ou aucun mode de verrouillage, voir Options de configuration de mappe dynamique.

Rubrique parent : Création de grilles de données simples

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
Configuration des mappes dynamiques
Configuration d'un expulseur TTL (Time To Live)
Java Configuration du cache local
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
Configuration d'une stratégie de verrouillage
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Configuration d'un expulseur TTL (Time To Live)

Lorsque vous créez une grille simple, une mappe par défaut (statique) et un ensemble de mappes dynamiques sont créés. Par défaut, aucun expulseur TTL n'est configuré pour une mappe par défaut. Si vous disposez d'une mappe dynamique, vous pouvez définir une valeur de durée de vie (TTL) pour la date/heure de création (.*CT), la date/heure de la dernière mise à jour (*.LUT) ou la date/heure du dernier accès (*.LAT). Vous pouvez modifier ce comportement par défaut de sorte qu'un expulseur TTL soit également activé pour une mappe par défaut.

Avant de commencer

- Créez une grille de données simple pour votre configuration.
- Décidez sur quel expulseur Time vous souhaitez l'utiliser. Pour plus d'informations sur les types d'expulseur spécifiques, voir Expulseurs.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Grille de données simple** > ***nom_grille_de_données*** > **Afficher les attributs avancés**.
2. Dans la zone de liste **Expulseur pour la mappe intitulée** , sélectionnez un type d'expulseur. Si vous décidez de conserver la valeur NONE pour cette option, l'expulseur TTL est désactivé uniquement pour une mappe par défaut. Dans le cas contraire, le fait de sélectionner un type d'expulseur autre que NONE active TTL pour une mappe par défaut.
3. Indiquez une valeur dans la zone **Expulser les données des mappes TTL (Time To Live) après**. La valeur par défaut TTL est de `3600` secondes. La valeur que vous indiquez ici s'applique également à une mappe par défaut si le type d'expulseur est paramétré sur une valeur autre que **NONE**. La valeur `0` permet de désactiver l'expulsion TTL pour la totalité de la grille.
4. Cliquez sur **Appliquer les modifications** pour enregistrer les modifications. Vous êtes averti de la perte de données une fois que la grille redémarre.

Exemple

Scénario par défaut :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **NONE** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `3600`. Il en résulte que les entrées sont expulsées au bout de 1 heure uniquement pour les mappes dynamiques se terminant par *.CT, *.LUT et *.LAT.

Scénario de personnalisation du type d'expulseur par défaut sur une valeur autre que NONE :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **Heure de création** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `3600`. Il en résulte que les entrées sont expulsées au bout de 1 heure pour les mappes dynamiques se terminant par *.CT, *.LUT et *.LAT. La mappe par défaut expulse les entrées 1 heure après l'heure de création.

Personnalisation du type d'expulseur par défaut et scénario de valeur TTL :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **Heure de la dernière mise à jour** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `4800`. Il en résulte que les entrées sont expulsées au bout de `80` minutes pour les mappes dynamiques se terminant par *.CT, *.LUT et *.LAT. La mappe par défaut expulse les entrées `80` minutes après l'heure de la dernière mise à jour de l'entrée.

Scénario de personnalisation de la valeur TTL uniquement :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **NONE** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `0`. Il en résulte que l'expulsion TTL est désactivée pour toutes les mappes (par défaut et dynamiques) de cette grille.

Scénario de personnalisation du type d'expulseur par défaut et de la valeur TTL :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **Heure du dernier accès** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `0`. Il en résulte que l'expulsion TTL est désactivée pour toutes les mappes (par défaut et dynamiques) de cette grille.

Expulseurs
Les expulseurs retirent des données de la grille de données. Vous pouvez configurer un expulseur pour une mappe dynamique et une mappe par défaut sur une grille simple.

Rubrique parent : Création de grilles de données simples

Concepts associés:
Expulseurs

Tâches associées:
Configuration des mappes dynamiques
Configuration d'une stratégie de verrouillage
Java Configuration du cache local
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs
Java 2.5+ Configuration de l'invalidation du cache local

Référence associée:
Options de configuration de mappe dynamique

Java

Expulseurs

Les expulseurs retirent des données de la grille de données. Vous pouvez configurer un expulseur pour une mappe dynamique et une mappe par défaut sur une grille simple.

Types d'expulseur

L'expulseur supprime les entrées en se basant sur un concept de durée de vie. Vous pouvez sélectionner un expulseur en fonction de l'heure de sa création, de l'heure de son dernier accès ou sa dernière mise à jour. Par défaut, un expulseur est créé à l'aide d'une mappe dynamique. Pour activer un expulseur sur une mappe par défaut pour une grille simple, voir Configuration d'un expulseur TTL (Time To Live).

Aucun

Spécifie que les entrées n'expirent jamais et par conséquent ne sont jamais supprimées de la mappe.

Heure de création

Spécifie que les entrées sont expulsées en fonction de la date et de l'heure auxquelles elles ont été créées.

Si vous utilisez l'attribut l'expulseur Heure de création , celui-ci expulse une entrée lorsqu'il aura atteint la durée de vie spécifiée par sa valeur TTL , qui est définie en millisecondes dans la configuration de l'application. Si vous paramétrez TTL sur la valeur de 10 secondes, l'entrée est automatiquement expulsée dix secondes après son insertion.

Il est important de faire attention lors de la définition de cette valeur pour le type d'expulseur Heure de création . Cet expulseur s'avère utile dans les cas de quantités raisonnablement élevées d'ajouts au cache utilisés pendant un temps donné. Avec cette stratégie, tout ce qui est créé est supprimé à la fin de la durée définie.

Le type d'expulseur Heure de création est utile dans des scénarios où l'on doit actualiser des cours boursiers toutes les 20 minutes, voire moins. Supposons qu'une application Web récupère les cours de la bourse, sans que l'obtention des cours les plus récents soit cruciale. Dans ce cas, les cours sont mis en cache dans une grille de données pendant 20 minutes. Après 20 minutes, les entrées de la mappe expirent et sont expulsées. Toutes les vingt minutes environ, la grille de données actualise les données à l'aide des données de la base de données. Celle-ci est actualisée toutes les 20 minutes avec les cours les plus récents.

Heure du dernier accès

Spécifie que les entrées sont expulsées en fonction de l'heure de leur dernier accès, qu'elles aient été lues ou actualisées.

Heure de la dernière modification

Spécifie que les entrées sont expulsées en fonction de l'heure de leur dernière modification.

Si vous utilisez le type d'expulseur Heure du dernier accès ou Heure de la dernière modification , paramétrez la valeur TTL sur une valeur plus faible que si vous utilisiez le type d'expulseur Heure de création . Les entrées sont réinitialisées lors de chaque accès. En d'autres termes, si la valeur de est égale à 15 et qu'une entrée a existé pendant 14 secondes mais qu'on y accède, elle n'expire pas de nouveau pendant les 15 prochaines secondes. Si vous paramétrez TTL sur une valeur relativement élevée, il est possible que de nombreuses entrées ne soient expulsées. Cependant, si vous le paramétrez sur une valeur telle que 15 secondes, les entrées avec peu d'accès risquent d'être supprimées.

Le type d'expulseur Heure du dernier accès ou Heure de la dernière mise à jour est utile dans des scénarios où l'on doit conserver les données de session d'un client, à l'aide d'une mappe de grille de données. Les données de session doivent être détruites si le client ne les utilise pas pendant un certain laps de temps. Par exemple, après 30 minutes d'inactivité du client. Dans ce cas, l'utilisation du type d'expulseur Heure du dernier accès ou Heure de la dernière mise à jour avec la valeur TTL paramétrée sur 30 minutes convient tout à fait à cette application.

Rubrique parent : Configuration d'un expulseur TTL (Time To Live)

Tâches associées:
Configuration d'un expulseur TTL (Time To Live)

Référence associée:
Options de configuration de mappe dynamique

Java

Configuration du cache local

Vous pouvez configurer vos clients pour un cache local en ligne L'on appelle cache local ce cache facultatif. Il s'agit d'une grille de données indépendante, présente sur chaque client et faisant office de cache du cache distant côté serveur. Le cache local est activé par défaut lorsque le verrouillage est désactivé ou configuré comme optimiste et ne peut pas être utilisé lorsqu'il est défini comme pessimiste.

Avant de commencer

Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.

Pourquoi et quand exécuter cette tâche

Un cache local est rapide, car il offre un accès en mémoire à un sous-ensemble de l'ensemble des données en cache stockées à distance. Par défaut, le cache local côté client n'a pas de taille maximale et des erreurs de mémoire insuffisante peuvent se produire sur le client. Pour contrôler la taille du cache local, configurez un remplacement côté client qui active un expulseur TTL ou LRU sur le client. Pour plus d'informations sur le remplacement des paramètres client, voir Configuration des clients Java avec une configuration XML et Configuration des clients Java à l'aide d'un programme.

Procédure

Vous pouvez activer la création d'un cache local sur le client par la définition du nom de mappe dynamique. Vous devez définir la mappe dynamique pour avoir un verrouillage optimiste ou aucun verrouillage. Exemples de noms de mappe pour lesquelles un cache local est activé :

```
my_grid.NONE
my_grid.NONE.O
```

Pour plus d'informations, voir Configuration des mappes dynamiques. Pour connaître la liste des options de configuration de mappe dynamique, voir Options de configuration de mappe dynamique.

Résultats

Pour vérifier qu'un cache local est activé, exécutez la méthode BackingMap.isNearCacheEnabled() dans le client. Vous pouvez également rechercher le message CWOBJ1128I dans les fichiers journaux sur le client pour déterminer si le cache local est activé.

Java 2.5+ **Configuration de l'invalidation du cache local**

Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée dans la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Ce mécanisme est plus rapide que l'utilisation de l'expulsion TTL (time to live, durée de vie) dans le cache local.

Rubrique parent : Création de grilles de données simples

Tâches associées:
Configuration des mappes dynamiques
Configuration d'une stratégie de verrouillage
Configuration d'un expulseur TTL (Time To Live)
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs

Java

Configuration de l'invalidation du cache local

2.5+ Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée dans la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Ce mécanisme est plus rapide que l'utilisation de l'expulsion TTL (time to live, durée de vie) dans le cache local.

Avant de commencer

- Vous devez utiliser IBM eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).
- Vous devez utiliser un cache local. Pour déterminer si le cache local est activé, exécutez la méthode BackingMap.isNearCacheEnabled() dans le client. Pour plus d'informations sur la configuration du cache local, voir Configuration du cache local.
- Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.

Pourquoi et quand exécuter cette tâche

L'activation de l'invalidation dans le cache local fournit un ensemble plus précis de données de la grille de données distante, car le cache local est mis à jour lorsque les données distantes sont modifiées.

Procédure

2.5+ Créez une mappe dynamique qui inclut l'option d'invalidation du cache local. Pour plus d'informations, voir Configuration des mappes dynamiques. Examinez les exemples de mappes suivants, pour lesquels l'invalidation du cache local est activée :

```
my_grid.NCI
my_grid.CT.NCI
my_grid.LAT.NCI
my_grid.LUT.NCI
my_grid.NONE.NCI
my_grid.CT.O.NCI
my_grid.LAT.O.NCI
my_grid.LUT.O.NCI
my_grid.NONE.O.NCI
```

Pour connaître la liste des options de configuration de mappe dynamique, voir Options de configuration de mappe dynamique.

Rubrique parent : Java Configuration du cache local

Tâches associées:
Configuration des mappes dynamiques
Configuration d'un expulseur TTL (Time To Live)

Référence associée:
Options de configuration de mappe dynamique

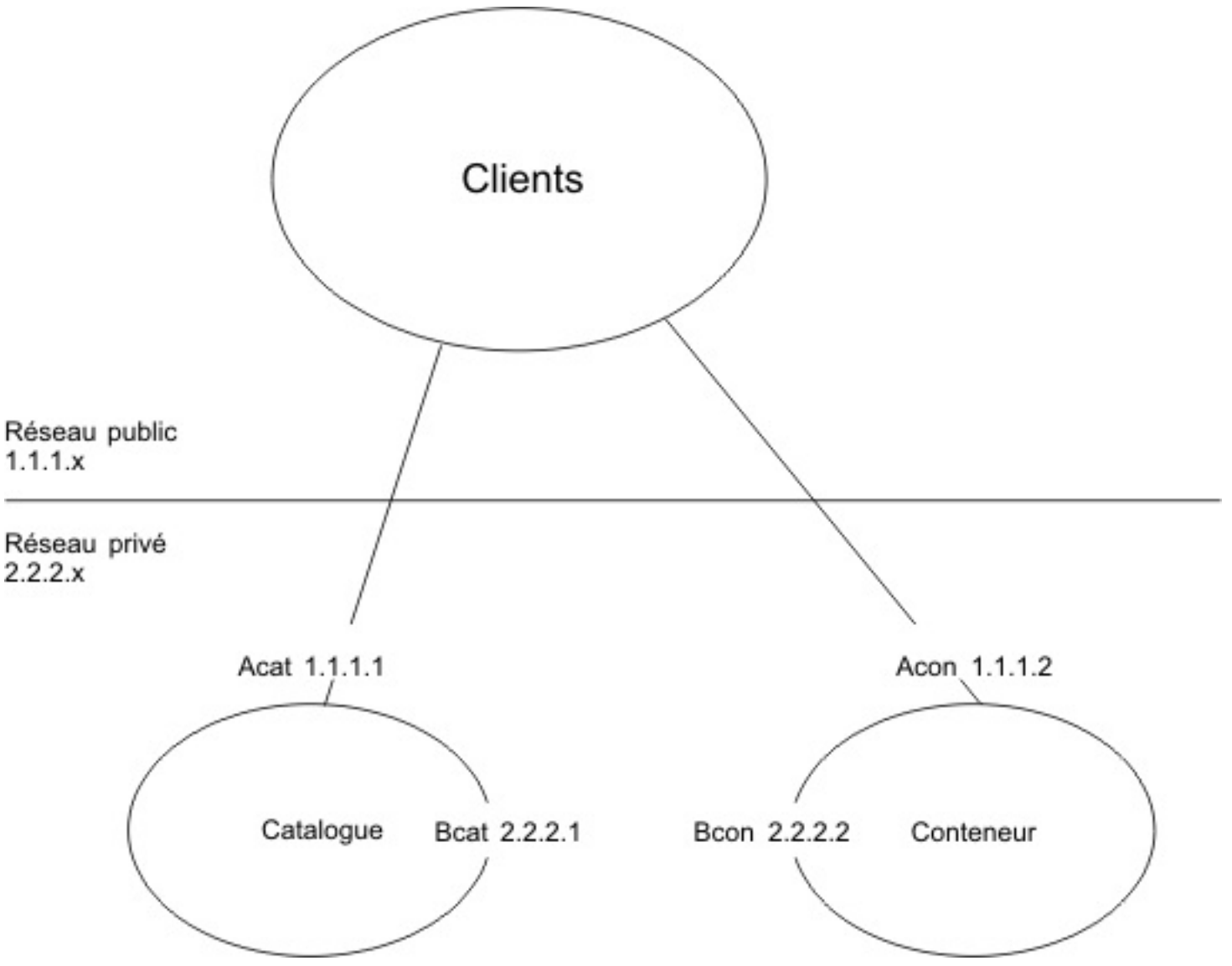
Gestion des noms d'hôte pour la communication entre les clients et les serveurs

Utilisez la propriété **`publishHost`** pour configurer le nom d’hôte que le transport par réseau publie à destination des serveurs et clients connectés.

Pourquoi et quand exécuter cette tâche

Avec le serveur DNS ou le fichier hosts, vous pouvez configurer et redéfinir l'hôte qui est publié à l'aide de la propriété **`publishHost`**. Vous disposez désormais d'un contrôle plus fin sur la communication entre les clients et les serveurs, ce qui vous permet également d'optimiser la communication entre les noeuds dans la grille de données. A titre d'exemple, le diagramme ci-dessous montre comment vous pouvez connecter les clients au protocole IP public (1.1.1.x), et les serveurs au protocole IP privé (2.2.2.x). Dans cet exemple, les serveurs de catalogue et de conteneur ont deux noms d'hôte pour chaque carte réseau : `A*` pour les adresses IP publiquement disponibles et `B*` pour les adresses IP privées.

*Figure 1. Topologie pour la propriété **`publishHost`**.*



Le nom d'hôte `C*` représente la nouvelle valeur de la propriété **`publishHost`**, qui est utilisée comme nom d'hôte générique à publier. Il constitue également une alternative à `A*` ou `B*`, car les noms `A*` ne sont pas appropriés pour les serveurs qui doivent communiquer sur un réseau privé et les noms `B*` ne sont pas appropriés pour les clients qui doivent communiquer sur un réseau public. Il vous permettra de choisir entre les adresses alternatives publique et privée des serveurs.

Procédure

1. Définissez la propriété **`publishHost`** avec la valeur `Ccat` sur le serveur de catalogue et avec la valeur `Ccon` sur le serveur de conteneur. Cette propriété définit le nom d'hôte qui est publié sur le DNS et pendant les communications des clients et des serveurs.
2. Facultatif : Définissez la propriété **`listenerHost`**. Cette propriété contrôle les adresses IP que chaque serveur écoute.
3. Assurez-vous que le DNS (ou les fichiers `/etc/hosts`) des clients qui utilisent le réseau public fait bien référence aux hôtes `Ccat` et `Ccon` à l'aide de leur adresse IP publique (respectivement Ccat 1.1.1.1 et Ccon 1.1.1.2). Assurez-vous que le DNS (ou les fichiers `/etc/hosts`) des serveurs qui utilisent le réseau privé fait bien référence aux hôtes `Ccat` et `Ccon` à l'aide de leur adresse IP privée (respectivement Ccat 2.2.2.1 et Ccon 2.2.2.2).
4. Utilisez la valeur **`publishHost`** du serveur de catalogue (Ccat) comme nom d'hôte des noeuds finaux de serveur de catalogue qui sont utilisés dans la configuration de grille d'objets sur les clients et les serveurs.

Résultats

Désormais, le trafic inter-serveur circule sur le réseau public, et le trafic entre les clients et les conteneurs circule sur le réseau privé.

Rubrique parent : Création de grilles de données simples

Tâches associées:
Configuration des mappes dynamiques
Configuration d'une stratégie de verrouillage
Configuration d'un expulseur TTL (Time To Live)
Java Configuration du cache local

Création de grilles de données de session

Vous pouvez créer des grilles de données pour enregistrer des sessions HTTP à partir de vos applications Java ou .NET.

Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données
Configurez votre application WebSphere Application Server pour qu'elle utilise le dispositif ou la gestion de session. Vous pouvez soit sélectionner le dispositif lorsque vous installez une nouvelle application, soit actualiser votre application existante ou les paramètres du serveur pour qu'ils utilisent le dispositif.

Configuration du gestionnaire de sessions HTTP avec WebSphere Application Server
Alors que WebSphere Application Server offre une fonction de gestion de sessions, les performances se dégradent quand le nombre de demandes augmente. WebSphere eXtreme Scale est fourni avec une mise en oeuvre de la gestion de sessions qui offre la réplication de session, une haute disponibilité, une meilleure évolutivité et des options de configuration plus robustes.

Configuration du gestionnaire de sessions HTTP avec WebSphere Portal
Vous pouvez rendre persistantes des sessions HTTP à partir de WebSphere Portal dans une grille de données.

Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications
WebSphere eXtreme Scale Client est fourni avec une implémentation de gestion de session qui remplace le gestionnaire de sessions par défaut pour un conteneur Web. Cette implémentation offre la réplication de session, la haute disponibilité, une meilleure évolutivité et des options de configuration. Vous pouvez activer le gestionnaire de réplication de session WebSphere eXtreme Scale Client et le démarrage du conteneur intégré ObjectGrid générique pour différents serveurs d'application, tels que Tomcat.

Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Vous pouvez migrer une session de réplication de mémoire à mémoire ou une session de base de données pour utiliser la gestion de session WebSphere eXtreme Scale.

.NET 2.5+ **Configuration d'un fournisseur de stockage d'état de session ASP.NET**
Vous pouvez stocker l'état de session de vos applications ASP.NET dans une grille de données.

2.5+ **Affichage de la taille des sessions HTTP**
Vous pouvez utiliser l'utilitaire **xscmd** pour afficher la taille des sessions dans votre application Java. Cette information peut vous aider à identifier et résoudre les problèmes qui peuvent survenir dans votre grille de données.

Paramètres d'initialisation du contexte de servlet
La liste de paramètres d'initialisation du contexte de servlet suivante peut être spécifiée dans le fichier splicer.properties en fonction de la méthode de raccord choisie.

Fichier splicer.properties
Le fichier `splicer.properties` contient toutes les options de configuration qui permettent de configurer un gestionnaire de sessions basé sur un filtre de servlet.

Rubrique parent : Configuration des grilles de données

Java

Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données

Configurez votre application WebSphere Application Server pour qu'elle utilise le dispositif ou la gestion de session. Vous pouvez soit sélectionner le dispositif lorsque vous installez une nouvelle application, soit actualiser votre application existante ou les paramètres du serveur pour qu'ils utilisent le dispositif.

Avant de commencer

Pour pouvoir modifier la configuration dans WebSphere Application Server, vous devez :

- disposer d'un accès à la cellule WebSphere Application Server à configurer ;
- connaître l'adresse IP ou le nom d'hôte qualifié complet du dispositif ;
- disposer d'un ID utilisateur et d'un mot de passe utilisables pour vous connecter à l'interface utilisateur du dispositif. Pour pouvoir créer une grille de données, vous devez disposer des droits de création d'une mémoire cache de données.
- avoir installé WebSphere eXtreme Scale Client dans la configuration WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client.
- avoir activé la sécurité globale dans la console d'administration WebSphere Application Server. Activez la sécurité globale si la sécurité de la couche de transport est activée sur le dispositif ou si vous voulez que les clients utilisent la sécurité de la couche de transport. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

Procédure

- **Pour configurer la gestion de session lors de l'installation de l'application, effectuez la procédure suivante :**

  1. Dans la console d'administration de WebSphere Application Server, cliquez sur **Applications** > **Nouvelle application** > **Nouvelle application d'entreprise**. Sélectionnez le chemin **Détaillé** pour la création de l'application, puis effectuez les premières étapes de l'assistant.
  2. A l'étape **Paramètres de gestion des sessions eXtreme Scale session** de l'assistant, configurez la grille de données que vous voulez utiliser. Pour la zone **Gérer la persistance des sessions par** , choisissez **WebSphere DataPower XC10 Appliance**. Entrez les informations concernant le dispositif et la grille de données du dispositif que vous voulez utiliser. Vous pouvez créer une grille de données ou utiliser une grille de données existante déjà configurée sur le dispositif.

     Si vous souhaitez sauvegarder vos sessions sur une grille de données existante du dispositif, vous devez connaître le nom de la grille de données à utiliser. Toutefois, vous pouvez créer une grille de données sur le dispositif lors de la configuration de l'application. Pour créer une grille de données de session avant de configurer l'application dans l'interface utilisateur du dispositif, cliquez sur **Grille de données** > **Session**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données de session à créer. Ce nom ne peut contenir les caractères suivants : `^ . \\ / , # $ @ : ; \ * ? < > | = + & % [ ] " "`.
  3. Terminez l'installation de l'application en effectuant la suite de procédure de l'assistant.

Vous pouvez également installer l'application à l'aide d'un script wsadmin. Dans l'exemple suivant, le paramètre**-SessionManagement** crée une configuration identique à celle que vous pouvez créer dans la console d'administration :

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission
.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XC10SessionManagement myXC10.ibm.com:!:username:!:password:!:AGrid80]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host]
[MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-
INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host]
[MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

- **Pour configurer la gestion de session sur une application existante dans la console d'administration de WebSphere Application Server :**

  Remarque : La case **Override session management** est cochée quand l'application est configurée pour utiliser WebSphere DataPower XC10 Appliance. Cela signifie que tout paramètre de session de niveau serveur défini dans la configuration de WebSphere Application Server sera remplacé par les paramètres de session de niveau application. Si ce n'est pas ce que vous souhaitez, vous pouvez activer WebSphere DataPower XC10 Appliance au niveau serveur.

  1. Dans la console d'administration WebSphere Application Server, cliquez sur **Applications** > **Types d'application** > **Applications d'entreprise WebSphere** > ***nom_application*** > **Propriétés du module Web** > **Gestion des sessions** > **Paramètres de gestion des sessions eXtreme Scale**.
  2. Actualisez les champs pour activer la persistance des sessions vers une grille de données.

  Vous pouvez également mettre à jour l'application à l'aide d'un script wsadmin. Dans l'exemple suivant, le paramètre**-SessionManagement** crée une configuration identique à celle que vous pouvez créer dans la console d'administration :

  ```
  AdminApp.edit('A-edition9.0', '[ -SessionManagement [[true
  XC10SessionManagement myXC10.ibm.com:!:username:!:password:!:AGrid80]]]')
  ```

  Les caractères `:!:` envoyés sont utilisés comme délimiteurs. Les valeurs transmises sont les suivantes :

  ```
  ID_application:!:nom_utilisateur:!:mot-de-passe:!:
  nom_grille
  ```

  Lorsque vous enregistrez les modifications, l'application utilise la grille de données configurée pour la persistance des sessions sur le dispositif.

- **Pour configurer la gestion de session sur un serveur existant :**

  1. Dans la console d'administration WebSphere Application Server, cliquez sur **Serveurs** > **types de serveur** > **Serveurs d'applications WebSphere** > ***nom_serveur*** > **Gestion des sessions** > **Paramètres de gestion des sessions eXtreme Scale**.
  2. Actualisez les champs pour activer la persistance des sessions.

  Les commandes wsadmin suivantes vous permettent de configurer également la gestion des sessions sur un serveur existant :

  ```
  AdminTask.configureServerSessionManagement('[-nodeName my_node
  -serverName server1 -enableSessionManagement true -sessionManagementType
  XC10SessionManagement -XC10SessionManagement [-applianceIdentifier myserver.ibm.com
  -userName -password ******** -gridName myTestGrid]]')
  ```

  Lorsque vous enregistrez les modifications, le serveur utilise la grille de données configurée pour la persistance de sessions avec toutes les applications qu'il exécute.

- Si vous souhaitez modifier d'autres aspects de la configuration des sessions HTTP, vous pouvez modifier le fichier `splicer.properties`. Vous pouvez obtenir l'emplacement du chemin du fichier `splicer.properties` en recherchant la propriété personnalisée **sessionFilterProps**. Si vous avez configuré la persistance de session au niveau du serveur, le nom de la propriété personnalisée est `com.ibm.websphere.xs.sessionFilterProps`. Si vous l'avez configurée au niveau de l'application, elle s'appelle *<nom_application>*`,com.ibm.websphere.xs.sessionFilterProps`. Ces propriétés personnalisées peuvent se trouver dans l'un des emplacements suivants :
  - Dans un environnement WebSphere Application Server Network Deployment : Le fichier `splicer.properties` se trouve dans le chemin du profil du gestionnaire de déploiement.
  - Dans un environnement WebSphere Application Server autonome : une propriété personnalisée sur le serveur d'applications

Vous pouvez ouvrir le fichier indiqué, effectuez les modifications et synchroniser les noeuds pour propager le fichier mis à jour des propriétés vers les autres noeuds de la configuration. Tous les noeuds de serveur d'applications nécessitent que le fichier `splicer.properties` se trouve dans le chemin défini pour que les sessions persistent.

Avertissement : Si vous souhaitez activer la persistence pour les sessions qui utilisent la réécriture d'URL, affectez à la propriété **`useURLEncoding`** la valeur `true` dans le fichier `splicer.properties`.

Pour plus d'informations sur les propriétés dans le fichier `splicer.properties`, voir Fichier splicer.properties.

Résultats

Vous avez configuré le gestionnaire de sessions HTTP pour que les sessions soient conservées vers une grille de données. Les entrées sont supprimées de la grille de données lorsque les sessions expirent. Voir Paramètres de gestion des sessions pour plus d'informations sur la mise à jour la valeur de temporisation des sessions dans la console d'administration WebSphere Application Server.

Si l'ensemble de la grille de données qui héberge les données de sessions d'application est inaccessible à partir du client du conteneur Web, le client utilise le conteneur Web de base dans la gestion de sessions WebSphere Application Server. La grille de données peut être inaccessible dans les cas suivants :

- Problème de réseau entre le conteneur Web et le dispositif.
- Arrêt des processus serveur dans le dispositif.

Les sessions les moins utilisées sont invalidés à partir du cache de session du conteneur Web. Si la grille de données dans le dispositif devient disponible, les sessions ayant été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données du dispositif est indisponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données de production lorsque le système est chargé.

ATTENTION :

Lorsque vous configurez ce scénario, les données d'identification de sécurité pour IBM WebSphere DataPower XC10 Appliance sont automatiquement stockées dans la configuration de WebSphere Application Server. Si vous êtes amené à modifier les données d'identification pour la grille de données après cette première configuration, WebSphere Application Server ne disposera plus des données d'identification correctes. Vous pourrez les réinitialiser en appliquant à nouveau les paramètres de gestion de session eXtreme Scale.

Que faire ensuite

- Configurez la sécurité avant de commencer à envoyer des données à la grille de données. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
- Configurez des répliques. Les répliques permettent de s'assurer que les données de vos grilles de données sont disponibles si la copie principale échoue. Pour configurer des répliques, cliquez sur **Grille de données** > **Session** > **Afficher les attributs avancés**. Les répliques ne sont créées que si le dispositif fait partie d'une collectivité. Si le nombre de dispositifs de la collectivité est *n*, le nombre maximal de répliques est *n*-*1*. Ainsi, si vous configurez trois répliques mais que vous n'avez que deux dispositifs dans la collectivité, une seule réplique est créée. Des répliques supplémentaires sont créées si vous ajoutez des dispositifs à la collectivité. Indiquez comme nombre de répliques le nombre idéal de répliques souhaitées : ainsi, lorsque des dispositifs rejoignent la collectivité, de nouvelles répliques sont créées. Le contenu de la grille de données est effacé lorsque vous modifiez le nombre de répliques.
- Configurez une limite de capacité pour la grille de données. En configurant des limites de capacité sur la grille de données, vous vous assurez que la capacité de stockage pour la collectivité est utilisée de façon prévisible. Pour plus d'informations, voir Configuration de la capacité maximale d'une grille de données.
- Vous pouvez surveiller votre grille de données de session dans l'interface utilisateur de DataPower XC10 Appliance. Pour plus d'informations, voir Surveillance des grilles de données dans l'interface utilisateur.

Rubrique parent : Création de grilles de données de session

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Référence associée:
Fichier splicer.properties

Information associée:
Installation de fichiers d'application d'entreprise à l'aide de la console
Installation d'applications d'entreprise à l'aide d'un script wsadmin

Java

Configuration du gestionnaire de sessions HTTP avec WebSphere Application Server

Alors que WebSphere Application Server offre une fonction de gestion de sessions, les performances se dégradent quand le nombre de demandes augmente. WebSphere eXtreme Scale est fourni avec une mise en oeuvre de la gestion de sessions qui offre la réplication de session, une haute disponibilité, une meilleure évolutivité et des options de configuration plus robustes.

Avant de commencer

- Par défaut, le fichier de propriétés du raccordeur (splicer) de XC10 utilise un ID utilisateur et un mot de passe générés par le système. Cet ID est XC10_USER_nomGrille et son mot de passe ne peut pas être modifié. Si vous changez l'ID utilisateur, vous devez manuellement mettre à jour la combinaison ID utilisateur/mot de passe dans le fichier `splicer.properties` en modifiant la propriété `credentialGeneratorProps`. Pour plus d'informations, voir Comment modifier le fichier splicer.properties. Cette combinaison générée par le système n'est utilisée pour obtenir le fichier `splicer.properties` que lors du démarrage, et ces données d'identification ne sont plus utilisées par le client WebSphere Application Server.
- WebSphere eXtreme Scale doit être installé dans votre cellule WebSphere Application Server ou WebSphere Application Server Network Deployment pour que vous puissiez utiliser le gestionnaire de sessions d'eXtreme Scale. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
- Lorsque vous utilisez WebSphere eXtreme Scale pour la réplication de session HTTP sur WebSphere Application Server, l'option **Autoriser la gestion des sessions de dépassement** doit être activée pour chaque application Web concernée et pour le serveur d'applications qui l'héberge. Pour plus d'informations, voir Paramètres de gestion de sessions.
- La sécurité globale doit être activée dans WebSphere Application Server si vous avez activé SSL (Secure Sockets Layer) sur le dispositif. Elle doit également l'être si vous voulez utiliser SSL pour une collectivité qui le prend en charge. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Le gestionnaire de sessions HTTP de WebSphere eXtreme Scale prend en charge les serveurs imbriqués et éloignés pour la mise en cache.

- **Scénario de serveurs imbriqués**

  Dans ce scénario, les serveurs de grille de données sont regroupés dans les processus où les servlets sont exécutés. Le gestionnaire de sessions peut communiquer directement avec l'instance ObjectGrid locale, pour éviter les retards réseau coûteux.

  Si vous utilisez WebSphere Application Server, placez dans les répertoires `META-INF` de vos fichiers d'archive Web (WAR) les fichiers *rép_base_wxs*`/session/samples/objectGrid.xml` et *rép_base_wxs* `/session/samples/objectGridDeployment.xml` fournis. eXtreme Scale détecte automatiquement ces fichiers au démarrage de l'application et démarre automatiquement les conteneurs eXtreme Scale dans le même processus que le gestionnaire de sessions.

  Vous pouvez modifier le fichier `objectGridDeployment.xml`, suivant que vous souhaitiez utiliser une réplication synchrone ou asynchrone et en fonction du nombre de répliques à configurer.

- **Scénario avec serveurs éloignés**

  Dans le scénario avec serveurs éloignés, les serveurs de conteneur et les servlets s'exécutent dans des processus différents. Le gestionnaire de sessions communique avec un serveur de conteneur éloigné. Pour pouvoir utiliser un serveur éloigné connecté au réseau, le gestionnaire de sessions doit être configuré avec les noms d'hôte et les numéros de port du domaine de service de catalogue. Le gestionnaire de sessions utilise ensuite une connexion client eXtreme Scale pour communiquer avec le serveur de catalogues et avec les serveurs de conteneur.

  Si les serveurs de conteneur démarrent dans des processus autonomes indépendants, démarrez les conteneurs de grille de données avec les fichiers `objectGridStandAlone.xml` et `objectGridDeploymentStandAlone.xml` fournis dans le répertoire des exemples du gestionnaire de sessions.

Procédure

1. Raccordez votre application de sorte qu'elle puisse utiliser le gestionnaire de sessions. Pour utiliser le gestionnaire de sessions, vous devez ajouter les déclarations de filtre appropriées aux descripteurs de

déploiement Web de l'application. En outre, les paramètres de configuration du gestionnaire de sessions sont transmis au gestionnaire de sessions sous la forme de paramètres d'initialisation du contexte de servlet dans les descripteurs de déploiement. Ces informations peuvent être injectées dans votre application de différentes manières :

- **Raccord automatique avec WebSphere Application Server**

  Lorsque vous installez votre application, vous pouvez la configurer pour qu'elle utilise le gestionnaire de sessions HTTP pour la grille de données. Vous pouvez également modifier la configuration de l'application ou du serveur pour qu'ils utilisent le gestionnaire WebSphere eXtreme Scale de sessions HTTP. Pour plus d'informations, voir Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données.

- **Raccord automatique de l'application avec des propriétés personnalisées**

  Vous n'avez pas besoin de raccorder manuellement vos applications lorsqu'elles s'exécutent dans WebSphere Application Server ou dans WebSphere Application Server Network Deployment.

  Ajoutez une propriété personnalisée à une cellule ou à un serveur pour définir à cette portée le fichier `splicer.properties` pour toutes les applications Web. Pour configurer la propriété personnalisée, procédez comme suit :

  a. Dans la console d'administration de WebSphere Application Server, accédez au chemin correct de l'emplacement que vous voulez définir dans la propriété personnalisée pour indiquer l'emplacement du fichier `splicer.properties`.
     - Pour définir la propriété personnalisée pour toutes les applications ou pour une application spécifique, cliquez sur **Administration du système** > **Cellule** > **Propriétés personnalisées**.
     - Pour définir la propriété personnalisée à appliquer à toutes les applications sur un serveur d'applications spécifique, cliquez sur **Serveur d'applications** > ***<nom_serveur>*** > **Administration** > **Propriétés personnalisées**. Le nom de la propriété est `com.ibm.websphere.xs.sessionFilterProps` et sa valeur se trouve dans le fichier `splicer.properties` dont a besoin votre application. Exemple de chemin d'emplacement d'un fichier : `/opt/splicer.properties`.
  b. Ajoutez la propriété personnalisée `com.ibm.websphere.xs.sessionFilterProps`. La valeur de cette propriété personnalisée indique l'emplacement du fichier `splicer.properties` à modifier. Le fichier existe dans le gestionnaire de déploiement. Si vous voulez indiquer le fichier `splicer.properties` pour une application spécifique à l'aide d'une propriété personnalisée au niveau de la cellule, entrez le nom de la propriété personnalisée comme `<`*`nom_application`*`>,com.ibm.websphere.xs.sessionFilterProps`, où *`nom_application`* représente le nom de l'application pour laquelle vous voulez appliquer la propriété personnalisée.

  Important : Vérifiez que le fichier `splicer.properties` mis à jour se trouve dans le même chemin sur tous les noeuds contenant un serveur d'applications hébergeant la ou les applications qui sont à raccorder pour la réplication de session.

  Les portées de cellule, de serveur et d'application sont les seules portées disponibles lors d'une exécution dans un gestionnaire de déploiement. Si vous avez besoin d'une autre portée, raccordez manuellement vos applications Web.

  A faire : Notez aussi que le raccordement automatique ne fonctionne que si tous les noeuds exécutant l'application contiennent le fichier `splicer.properties` dans le même chemin. Dans le cas d'environnements mixtes contenant des noeuds Windows et UNIX, cette manière de procéder n'est pas disponible et vous devez raccorder manuellement l'application.

- **Raccorder l'application avec le script `addObjectGridFilter`**

  Utilisez un script de ligne de commande fourni avec eXtreme Scale pour raccorder une application avec des déclarations de filtre et une configuration sous forme de paramètres d'initialisation de contexte de servlet. Pour un déploiement WebSphere Application Server, ce script se trouve dans `<répertoire_principal_was>/optionalLibraries/ObjectGrid/session/bin/addObjectGridFilter.bat/sh`. Pour un déploiement autonome, il se trouve dans `REP_PRINCIPAL_WXS/ObjectGrid/session/bin/addObjectGridFilter.sh/bat`. Le script **`oddObjectGridFilter`** utilise deux paramètres :
  - Application : chemin absolu du fichier archive d'entreprise à raccorder
  - Chemin absolu du fichier de propriétés du raccordeur qui contient des propriétés de configuration.

Voici le format du script :

Windows

```
addObjectGridFilter.bat [fichier_ear]
[fichier_propriétés_raccordeur]
```

UNIX

```
addObjectGridFilter.sh [fichier_ear]
[fichier_propriétés_raccordeur]
```

UNIX **Exemple utilisant eXtreme Scale installé sur WebSphere Application Server sur UNIX :**

a. `cd` *`rép_base_wxs`*`/optionalLibraries/ObjectGrid/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear`
   `racine_was``/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Exemple utilisant eXtreme Scale installé dans un répertoire autonome sur UNIX :**

a. `cd` *`racine_was`*`/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear`
   `racine_was``/session/samples/splicer.properties`

Le filtre de servlet qui est raccordé conserve les valeurs de configuration par défaut. Vous pouvez remplacer ces valeurs par défaut par des options de configuration que vous spécifiez dans le fichier de propriétés, dans le second argument. Pour une liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

Vous pouvez modifier et utiliser l'exemple de fichier `splicer.properties` fourni avec l'installation d'eXtreme Scale. Vous pouvez également utiliser le script **`addObjectGridServlets`**, qui insère le gestionnaire de sessions en étendant chaque servlet. Mais le script recommandé est le script **`addObjectGridFilter`**.

- **Raccorder manuellement l'application avec le script de génération Ant**

  WebSphere eXtreme Scale est fourni avec un fichier `build.xml` qui peut être utilisé par Apache Ant, qui est inclus dans le dossier *`racine_was`*`/bin` d'une installation WebSphere Application Server. Vous pouvez modifier ce fichier `build.xml` pour modifier les propriétés de configuration du gestionnaire de sessions. Le nom de ces propriétés de configuration est identique au nom des propriétés contenues dans le fichier `splicer.properties`. Pour modifier le fichier `build.xml`, appelez le processus Ant en exécutant la commande suivante :
  - UNIX `ant.sh`, `ws_ant.sh`
  - Windows `ant.bat`, `ws_ant.bat`

  (UNIX) ou (Windows).
- **Mettre à jour manuellement le descripteur Web**

  Editez le fichier `web.xml` fourni avec l'application Web pour incorporer la déclaration de filtre, son mappage de servlets et les paramètres d'initialisation du contexte de servlet. N'utilisez pas cette méthode car elle présente des risques d'erreurs.

Pour connaître la liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

2. Déployez l'application. Déployez l'application à l'aide de votre procédure normale pour un serveur ou un cluster. Une fois que vous avez déployé l'application, vous pouvez la démarrer.
3. Accédez à l'application. Vous pouvez maintenant accéder à l'application, qui interagit avec le gestionnaire de sessions et WebSphere eXtreme Scale.

Que faire ensuite

Vous pouvez modifier la plupart des attributs de configuration du gestionnaire de sessions lorsque vous instrumentez votre application pour utiliser le gestionnaire de sessions. Ces attributs sont : réplication synchrone ou asynchrone, taille de la table de session en mémoire, etc. En dehors des attributs modifiables lors de l'instrumentation de l'application, les seuls autres attributs de configuration que vous pouvez modifier après le déploiement de l'application sont ceux liés à la topologie des clusters de serveurs WebSphere eXtreme Scale et à la manière dont leurs clients (gestionnaires de sessions) s'y connectent.

Comportement de scénarios distant : si la grille de données complète qui héberge les données de sessions d'application est inaccessible à partir du client du conteneur Web, le client utilise le conteneur Web de base dans WebSphere Application Server pour la gestion des sessions. La grille de données peut être

inaccessible dans les cas suivants :

- Problème de réseau entre le conteneur Web et les serveurs de conteneur distants.
- Arrêt des processus serveur de conteneur distant.

Le nombre de références de session conservées en mémoire, spécifié par le paramètre **sessionTableSize**, est maintenu même lorsque les sessions sont stockées dans le conteneur Web de base. Les sessions les moins utilisées sont invalidées à partir du cache de session du conteneur Web lorsque la valeur **sessionTableSize** est dépassée. Si la grille de données distante devient disponible, les sessions qui ont été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données distante n'est pas disponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données distante de production lorsque le système est chargé.

Rubrique parent : Création de grilles de données de session

Java

Configuration du gestionnaire de sessions HTTP avec WebSphere Portal

Vous pouvez rendre persistantes des sessions HTTP à partir de WebSphere Portal dans une grille de données.

Avant de commencer

Votre environnement WebSphere eXtreme Scale Client et WebSphere Portal doivent satisfaire aux exigences suivantes :

- Installez WebSphere eXtreme Scale Client sur vos noeuds WebSphere Application Server et WebSphere Portal. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
- WebSphere Portal version 7 ou suivante.
- Les portlets personnalisés doivent être configurés dans WebSphere Portal. Les portlets d'administration livrés avec WebSphere Portal ne peuvent actuellement pas être intégrés avec des grilles de données.

Pourquoi et quand exécuter cette tâche

L'introduction de WebSphere DataPower XC10 Appliance dans un environnement WebSphere Portal peut être bénéfique dans les cas suivants :

Important : Bien que les scénarios suivants présentent des avantages, l'introduction de WebSphere DataPower XC10 Appliance dans l'environnement peut entraîner une utilisation plus importante des processeurs au niveau de WebSphere.

- **Lorsque la persistance des sessions est requise.**

  Par exemple, si les données de session de vos portlets personnalisés doivent rester disponibles lors d'une défaillance de WebSphere Portal Server, vous pouvez rendre persistantes les sessions HTTP dans la grille de données WebSphere DataPower XC10 Appliance. Les données sont répliquées entre de nombreux serveurs, accroissant la disponibilité des données.

- **Dans une topologie avec plusieurs centres de données.**

  Si votre topologie couvre plusieurs centres de données à travers différents emplacements physiques, vous pouvez rendre persistantes les sessions HTTP de WebSphere Portal dans la grille de données WebSphere DataPower XC10 Appliance. Les sessions sont répliquées dans les grilles de données des centres de données. Si un centre de données est défaillant, les sessions sont basculées vers un autre centre de données qui contient une copie des données de la grille de données.

- **Pour diminuer la mémoire requise au niveau de WebSphere Portal Server.**

  En déchargeant les données de session sur un groupe de serveurs de conteneurs, un sous-ensemble des sessions se trouve sur les serveurs WebSphere Portal. Ce déchargement de données réduit la mémoire requise au niveau de WebSphere Portal Server.

Procédure

1. Configurez l'application WebSphere Portal `wps` et les éventuels portlets personnalisés pour permettre aux sessions d'être stockées dans la grille de données.

   Pour plus d'informations, voir Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données. Cet action aboutit au raccordement des portlets personnalisés pour permettre la persistance de session sur votre grille de données.

2. Si TLS/SSL (Transport Layer Security/Secure Sockets Layer) est configuré pour le serveur WebSphere Portal et pour le dispositif, vous devez configurer les fichiers de clés certifiées TLS/SSL.
   - Si la communication sortante résultante du serveur WebSphere Portal vers le dispositif utilise TLS/SSL, vous devez ajouter le certificat du dispositif à la configuration de WebSphere Application Server. Utilisez le script **`addXC10PublicCert.py`**. Il se trouve dans le répertoire *`racine_was`*`/bin` :

     ```
     wsadmin.bat -conntype SOAP -port <PORT_SOAP_PORTAL_SERVER> -lang jython -user
     wpsadmin -password wpsadmin -f addXC10PublicCert.py
     ```

   - Si la communication entrante résultante du dispositif vers le serveur WebSphere Portal utilise TLS/SSL, mettez à jour le fichier de clés certifiées du dispositif pour inclure les certificats publics pour le serveur WebSphere Portal. La mise à jour du fichier de clés certifiées permet la communication entre le dispositif et WebSphere Portal.

a. Extrayez la clé publique du certificat personnel Portal Server. Utilisez l'utilitaire **IKEYMAN**. Cet utilitaire crée un fichier `.arm`. Pour plus d'informations, voir Extraction de certificats publics pour les fichiers de clés certifiées.
b. Téléchargez le fichier de clés certifiées public pour le dispositif. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..
c. Utilisez l'utilitaire **iKeyman** pour mettre à jour le fichier `truststore.jks` que vous avez extrait du dispositif avec le certificat Portal Server public dans le fichier `.arm`. Pour plus d'informations, voir Importation de certificats de signataire.
d. Téléchargez le fichier de clés certifiées mis à jour vers le dispositif. Cliquez sur **Soumettre les paramètres TLS** après avoir téléchargé le fichier de clés certifiées. La collectivité redémarre automatiquement lorsque vous soumettez les paramètres TLS et le nouveau fichier de clés certifiées est ajouté aux autres dispositifs de la collectivité. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

3. Certaines versions du serveur WebSphere Portal peuvent présenter des erreurs d'exécution lorsque des cookies sont ajoutés à une réponse HTTP. Puisque WebSphere DataPower XC10 Appliance ajoute des cookies pour permettre le basculement et à d'autres fins, ces cookies doivent être ajoutés à la liste des cookies à ignorer du serveur WebSphere Portal. Pour plus d'informations, voir la section décrivant le paramètre cookie.ignore.regex de la rubrique consacrée à la mise en cache des pages partagées par plusieurs utilisateurs, sur le wiki dédié à IBM WebSphere Portal. Les deux cookies qui doivent être ajoutés à la liste sont IBMID.* et IBMSessionHandle.*. La liste mise à jour peut ressembler à ceci : `"digest\\.ignore.*|LtpaToken|LtpaToken2|JSESSIONID|IBMID.*|IBMSessionHandle.*"`. Pour plus d'informations, voir la rubrique consacrée à la mise en cache des pages partagées par plusieurs utilisateurs sur le wiki dédié à IBM WebSphere Portal.
4. Redémarrez les serveurs WebSphere Portal. Pour plus d'informations, voir WebSphere Portal version 7 : Démarrage et arrêt des serveurs, des gestionnaires de déploiement et des agents de noeud.

Résultats

Vous pouvez accéder à WebSphere Portal Server ; les données de session HTTP pour les portlets personnalisés configurés sont conservées dans la grille de données.

Si l'ensemble de la grille de données qui héberge les données de sessions d'application est inaccessible à partir du client de conteneur Web, le client utilise le conteneur Web de base dans la gestion de sessions WebSphere Application Server. La grille de données peut être inaccessible dans les cas suivants :

- Problème de réseau entre le conteneur Web et les serveurs de conteneur distants.
- Arrêt des processus serveur de conteneur distant.

Le nombre de références de session conservées en mémoire, spécifié par le paramètre **sessionTableSize**, est maintenu même lorsque les sessions sont stockées dans le conteneur Web de base. Les sessions les moins utilisées sont invalidées à partir du cache de session du conteneur Web lorsque la valeur **sessionTableSize** est dépassée. Si la grille de données distante devient disponible, les sessions ayant été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données distante n'est pas disponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données distante de production lorsque le système est chargé.

Rubrique parent : Création de grilles de données de session

Tâches associées:
Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications
Administration à l'aide de l'interface de commande HTTP

Information associée:
Gestion des clés avec l'interface graphique IKEYMAN

Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications

WebSphere eXtreme Scale Client est fourni avec une implémentation de gestion de session qui remplace le gestionnaire de sessions par défaut pour un conteneur Web. Cette implémentation offre la réplication de session, la haute disponibilité, une meilleure évolutivité et des options de configuration. Vous pouvez activer le gestionnaire de réplication de session WebSphere eXtreme Scale Client et le démarrage du conteneur intégré ObjectGrid générique pour différents serveurs d'application, tels que Tomcat.

Avant de commencer

Vous pouvez activer Transport Layer Security (TLS) avec SSL sur WebSphere DataPower XC10 Appliance. Pour plus d'informations, voir Configuration de TLS pour les applications de grille de données.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser le gestionnaire de sessions HTTP avec d'autres serveurs d'applications qui n'exécutent pas WebSphere Application Server, tels que WebSphere Application Server Community Edition. Pour configurer d'autres serveurs d'applications pour qu'ils utilisent la grille de données, vous devez raccorder votre application et incorporer des fichiers WebSphere eXtreme Scale Client Java archive (JAR) dans votre application.

Procédure

1. Raccordez votre application de sorte qu'elle puisse utiliser le gestionnaire de sessions. Pour utiliser le gestionnaire de sessions, vous devez ajouter les déclarations de filtre appropriées aux descripteurs de déploiement Web de l'application. En outre, les paramètres de configuration du gestionnaire de sessions sont transmis au gestionnaire de sessions sous la forme de paramètres d'initialisation du contexte de servlet dans les descripteurs de déploiement. Vous disposez de trois manières de présenter ces informations dans votre application :
   - Script **`addObjectGridFilter`** :

     Utilisez un script de ligne de commande fourni avec WebSphere eXtreme Scale Client pour raccorder une application avec des déclarations de filtre et une configuration sous forme de paramètres d'initialisation de contexte de servlet. Le script *rép_base_wxs*`/session/bin/addObjectGridFilter.sh|bat` accepte deux paramètres : le chemin absolu d'accès au fichier EAR (enterprise archive) ou au fichier WAR (web archive) à raccorder et le chemin absolu au fichier des propriétés splicer qui contient diverses propriétés de configuration. La syntaxe de ce script est la suivante :

     Windows

     ```
     addObjectGridFilter.bat <fichier_ear_ou_war> <fichier_propriétés_splicer>
     ```

     UNIX

     ```
     addObjectGridFilter.sh
     <fichier_ear_ou_war> <fichier_propriétés_splicer>
     ```

     UNIX Exemple d'utilisation de WebSphere eXtreme Scale Client installé dans un répertoire autonome sur UNIX :

     a. `cd` *rép_base_wxs*`/session/bin`
     b. `addObjectGridFilter.sh /tmp/mySessionTest.ear` *rép_base_wxs*`/session/samples/splicer.properties`

     Le filtre de servlet qui est joint conserve les valeurs de configuration par défaut. Vous pouvez remplacer ces valeurs par défaut par des options de configuration que vous spécifiez dans le fichier de propriétés, dans le second argument. Pour connaître la liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

     Vous pouvez modifier et utiliser l'exemple de fichier `splicer.properties` fourni avec l'installation d'WebSphere eXtreme Scale Client. Toutefois, vous devez examiner le fichier afin de vous assurer qu'aucune des propriétés que vous voulez utiliser ne figure en commentaire (comme, par exemple, la propriété `catalogHostPort`). Vous pouvez également utiliser le script **`addObjectGridServlets`**, qui insère le gestionnaire de sessions en étendant chaque servlet. Cependant, le script recommandé est le script **`addObjectGridFilter`**.

   - Script de génération Ant :

     WebSphere eXtreme Scale Client est fourni avec un fichier `build.xml` qui peut être utilisé par

Apache Ant, qui est inclus dans le dossier *racine_was*/bin d'une installation WebSphere Application Server. Vous pouvez modifier le fichier build.xml pour changer les propriétés de configuration du gestionnaire de sessions. Les propriétés de configuration correspondent aux noms de propriété dans le fichier splicer.properties. Une fois que le fichier build.xml a été modifié, appelez le processus Ant en exécutant ant.sh, ws_ant.sh (UNIX) ou ant.bat, ws_ant.bat (Windows).

- Mise à jour manuelle du descripteur Web :

  Editez le fichier web.xml qui est packagé avec l'application Web pour incorporer la déclaration de filtre, son mappage de servlets et les paramètres d'initialisation du contexte de servlet. N'utilisez pas cette méthode car elle est source d'erreurs possibles.

Pour connaître la liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

2. Incorporez dans votre application les fichiers JAR du gestionnaire de réplication de sessions d'WebSphere eXtreme Scale Client. Vous pouvez incorporer les fichiers dans le répertoire WEB-INF/lib des modules d'application ou dans le chemin d'accès aux classes du serveur d'applications. Les fichiers JAR requis varient selon le type de conteneurs utilisés :
   - Serveurs de conteneur distants : ogclient.jar et sessionobjectgrid.jar
   - Serveurs de conteneur intégrés : objectgrid.jar et sessionobjectgrid.jar
3. Déployez l'application. Déployez l' application à l'aide de votre procédure normale pour un serveur ou un cluster. Une fois l'application déployée, vous pouvez la démarrer.
4. Accédez à l'application. Vous pouvez maintenant accéder à l'application, qui interagit avec le gestionnaire de sessions et WebSphere eXtreme Scale Client.

Que faire ensuite

Vous pouvez modifier la majorité des attributs de configuration du gestionnaire de sessions lorsque vous instrumentez votre application pour utiliser le gestionnaire de sessions. Ces attributs sont des variantes du type de réplication (synchrone ou asynchrone), la taille de la table des sessions en mémoire, etc. En dehors des attributs modifiables lors de l'instrumentation de l'application, les seuls autres attributs de configuration que vous pouvez modifier après le déploiement de l'application sont ceux liés à la topologie des clusters de serveurs WebSphere eXtreme Scale et à la manière dont leurs clients (gestionnaires de sessions) s'y connectent.

Comportement dans le scénario distant : si l'ensemble de la grille de données qui héberge les données de session d'application est inaccessible depuis le client du conteneur Web, le client utilise à la place le conteneur Web de base du serveur d'applications pour gérer les sessions. La grille de données peut être inaccessible dans les scénarios suivants :

- Problème de réseau entre le conteneur Web et les serveurs de conteneur distants.
- Arrêt des processus serveur de conteneur distant.

Le nombre de références de session conservées en mémoire, spécifié par le paramètre **sessionTableSize**, est maintenu même lorsque les sessions sont stockées dans le conteneur Web de base. Les sessions les moins utilisées sont invalidées à partir du cache de session du conteneur Web lorsque la valeur **sessionTableSize** est dépassée. Si la grille de données distante devient disponible, les sessions ayant été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données distante n'est pas disponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données distante de production lorsque le système est chargé.

Rubrique parent : Création de grilles de données de session

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Tâches associées:
Configuration du gestionnaire de sessions HTTP avec WebSphere Portal
Administration à l'aide de l'interface de commande HTTP

Référence associée:
Fichier splicer.properties

Information associée:
Installation de fichiers d'application d'entreprise à l'aide de la console
Installation d'applications d'entreprise à l'aide d'un script wsadmin
Gestion des clés avec l'interface graphique IKEYMAN

.NET

Configuration d'un fournisseur de stockage d'état de session ASP.NET

2.5+ Vous pouvez stocker l'état de session de vos applications ASP.NET dans une grille de données.

Avant de commencer

- Installez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Présentation de l'installation de WebSphere eXtreme Scale Client for .NET.
- Votre environnement .NET doit respecter la configuration système requise. Pour plus d'informations, voir Remarques relatives à Microsoft .NET.
- Votre application .NET doit être configurée pour gérer l'état de session.

.NET 2.5+ **Création d'une grille de données à utiliser avec le fournisseur de stockage d'état de session ASP.NET**
Créez une grille de données pour enregistrer l'état de session de vos applications ASP.NET.

.NET 2.5+ **Configuration de votre application .NET application pour l'utilisation du fournisseur de stockage d'état de session ASP.NET**
Pour configurer le fournisseur de stockage d'état de session ASP.NET, vous devez mettre à jour le fichier `web.config` de l'application ASP.NET en y définissant le fournisseur de stockage d'état de session ASP.NET et sa configuration.

Rubrique parent : Création de grilles de données de session

Création d'une grille de données à utiliser avec le fournisseur de stockage d'état de session ASP.NET

2.5+ Créez une grille de données pour enregistrer l'état de session de vos applications ASP.NET.

Procédure

Créez une grille de données pour votre fournisseur de stockage d'état de session. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Session**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données de session à créer. Par défaut, le fournisseur de stockage d'état de session ASP.NET est configuré pour utiliser une grille nommée `session`. Si vous créez une grille avec un nom différent, configurez le fournisseur avec ce nom.

Que faire ensuite

Mettez à jour la configuration de votre application Web afin que cette dernière utilise la grille de données. Pour plus d'informations, voir Configuration de votre application .NET application pour l'utilisation du fournisseur de stockage d'état de session ASP.NET.

Rubrique parent : .NET 2.5+ Configuration d'un fournisseur de stockage d'état de session ASP.NET

.NET

Configuration de votre application .NET application pour l'utilisation du fournisseur de stockage d'état de session ASP.NET

2.5+ Pour configurer le fournisseur de stockage d'état de session ASP.NET, vous devez mettre à jour le fichier `web.config` de l'application ASP.NET en y définissant le fournisseur de stockage d'état de session ASP.NET et sa configuration.

Avant de commencer

- Configurez une grille de données pour le stockage de l'état de session HTTP ASP.NET. Pour plus d'informations, voir Création d'une grille de données à utiliser avec le fournisseur de stockage d'état de session ASP.NET.
- Vous devez connaître l'hôte et le port de serveur de catalogue. Pour obtenir l'hôte et le port du serveur de catalogue dans l'interface utilisateur du dispositif, cliquez sur **Collectivité** > **Membres**. Sélectionnez un membre de collectivité. L'adresse IP et le numéro de port du serveur de catalogue s'affichent.

Procédure

1. Mettez à jour le fichier `web.config` de votre application ASP.NET en y définissant les paramètres du fournisseur de stockage d'état de session ASP.NET. Vous devez mettre à jour ou ajouter dans le fichier `web.config` le texte qui figure en gras dans l'exemple suivant :

```
<system.web>
        …
        <sessionState
                mode="Custom"
                customProvider="WxsSessionStateStoreProvider">
                <providers>
                        <add
                                name="WxsSessionStateStoreProvider"
type="IBM.WebSphere.Caching.SessionStateStore.WxsSessionStateStore,
                                        IBM.WebSphere.Caching,
Version=8.6.0.2, Culture=neutral,
                                        PublicKeyToken=b439a24ee43b0816"
                                wxsPropertyFile="\optional\path\to\NET-
client.properties"
                                wxsHostAndPort="optionalHostAndPort"
                                wxsGridName="session"
                                wxsMapName="ASPNET.SessionState"

         />
        </providers>
                </sessionState>
        …
</system.web>
```

wxsPropertyFile (facultatif)

Indique le nom entièrement qualifié du fichier de propriétés que le fournisseur utilise lorsqu'il se connecte à la grille de données via l'API Connect. Si cet attribut n'est pas spécifié ou contient une chaîne vide, le fournisseur recherche le fichier `Client.Net.properties` dans le répertoire d'exécution en cours du processus de l'application Web. Si le fournisseur ne trouve pas ce fichier dans ce répertoire, il le recherche dans le répertoire *net_client_home*`\config`.

wxsHostAndPort

Indique la liste séparée par des virgules des paires hôte et port correspondant aux serveurs de catalogue auxquels le fournisseur de stockage d'état de session se connecte lorsqu'il accède à la grille de données. Le format est le suivant :

```
<nom ou adresse IP de l'hôte>:<port tcp>[,<nom ou adresse IP de l'hôte>:<port tcp>]
```

wxsGridName (facultatif)

Indique le nom de la grille de données à laquelle le fournisseur de stockage de session ASP.NET se connecte. Si vous avez créé une grille de données pour les états de session ASP.NET, indiquez son nom. Si vous n'indiquez pas de valeur, le fournisseur se connecte à la grille de données `session`.

wxsMapName (facultatif)

Indique la mappe à laquelle le fournisseur se connecte. Si vous n'indiquez pas de valeur, le fournisseur se connecte à la mappe `ASPNET.SessionState`.

2. Redémarrez l'application Web cible. L'application Web doit redémarrer afin qu'IIS puisse charger le fournisseur. Dans la plupart des cas, une fois le fichier `web.config` modifié et le traitement de la demande HTTP en cours terminé, le redémarrage s'effectue automatiquement.

Résultats

L'état de session ASP.NET de votre application ASP.NET est stocké dans la grille de données.

Rubrique parent : .NET 2.5+ Configuration d'un fournisseur de stockage d'état de session ASP.NET

Java

Affichage de la taille des sessions HTTP

2.5+ Vous pouvez utiliser l'utilitaire **xscmd** pour afficher la taille des sessions dans votre application Java. Cette information peut vous aider à identifier et résoudre les problèmes qui peuvent survenir dans votre grille de données.

Avant de commencer

- L'utilitaire **xscmd** doit être configuré pour se connecter à vos serveurs de catalogue. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.
- Vous devez connaître l'ID et le descripteur de la session dont vous voulez afficher la taille.

Procédure

Exécutez la commande **xscmd -c showSessionSize**. La syntaxe de cette commande est la suivante :

```
xscmd.bat|sh -c showSessionSize   -cep nom_hôte:port(,nom_hôte:port)
-sid ID_session -sh descripteur_session -g nom_grille
[-wacr racine_contexte_application_web]
```

Par exemple, vous pouvez exécuter la commande suivante :

```
xscmd.bat -c showSessionSize -g session -cep 9.42.139.213:2809 -sh c -sid
LFMzQnDX5k87xztMF3ri6jU -wacr /A -user xcadmin -pwd xcadmin
```

Cette commande affiche les informations de session suivantes :

```
*** ID session : LFMzQnDX5k87xztMF3ri6jU
  Nombre d'attributs de session : 2
  Taille totale des métadonnées de session : 488 octets
  Taille totale des attributs de session : 243 octets
  Taille totale de session : 731 octets

  Clé et valeur de métadonnées :
  Clé : LFMzQnDX5k87xztMF3ri6jU/A
  Taille de la clé : 88 octets
  Taille de la valeur : 400 octets

  Clé et valeur d'attribut individuel :
  Clé : LFMzQnDX5k87xztMF3ri6jU/A_session.reqCount
  Taille de la clé : 128 octets
  Taille de la valeur : 2 octets
  Clé : LFMzQnDX5k87xztMF3ri6jU/A_user
  Taille de la clé : 104 octets
  Taille de la valeur : 9 octets
```

Rubrique parent : Création de grilles de données de session

Paramètres d'initialisation du contexte de servlet

La liste de paramètres d'initialisation du contexte de servlet suivante peut être spécifiée dans le fichier splicer.properties en fonction de la méthode de raccord choisie.

Paramètres

`applicationQualifiedCookies`

Valeur de type chaîne `true` ou `false`. Utilisez la valeur `true` si votre environnement contient plusieurs applications qui utilisent des noms de cookie uniques. La valeur par défaut est `false`, ce qui suppose que toutes les applications utilisent le même nom de cookie.

`authenticationRetryCount`

Indique le nombre de tentatives d'authentification qui ont lieu lorsque les données d'identification ont expiré. Si la valeur est 0, aucune nouvelle tentative d'authentification n'a lieu.

`catalogHostPort`

Le serveur de catalogues peut être contacté pour obtenir une instance ObjectGrid côté client. La valeur doit avoir le format `hôte:port<,hôte:port>`. L'hôte est l'hôte d'écoute sur lequel le serveur de catalogue s'exécute. Le port est le port d'écoute du processus serveur de catalogue. Cette liste peut avoir une longueur quelconque et n'est utilisée que pour l'amorçage. La première adresse viable est utilisée. Elle est facultative dans WebSphere Application Server si la propriété **`catalog.services.cluster`** est définie.

`credentialAuthentication`

Indique la prise en charge de l'authentification des données d'identification du client. Les valeurs possibles sont :

- Jamais : le client ne prend pas en charge l'authentification des données d'identification.
- Pris en charge : le client prend en charge l'authentification des données d'identification si et seulement si le serveur en fait de même.
- Obligatoire : le client requiert l'authentification des données d'identification. La valeur par défaut est Pris en charge.

`cookieDomain`

Spécifie si vous exigez que les sessions soient accessibles à travers les hôtes. Définissez la valeur avec le nom du domaine commun entre les hôtes.

`cookiePath`

Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe est utilisée pour obtenir les données d'identification des clients.

`credentialGeneratorClass`

Le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe sert à obtenir les données d'identification des clients.

`credentialGeneratorProps`

Les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés sont affectées à l'objet à l'aide de la méthode setProperties(String). La valeur credentialGeneratorProps n'est utilisée que si la valeur de la propriété **`credentialGeneratorClass`** n'est pas NULL.

`enableSessionStats`

Valeur de type chaîne `true` ou `false`. Active le suivi des statistiques des sessions HTTP client eXtreme Scale.

`fragmentedSession`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `true`. Ce paramètre permet de contrôler si le produit stocke les données de session en tant qu'entrée d'un seul bloc ou s'il stocke chaque attribut séparément.

Utilisez la valeur `true` si la session d'application Web a de nombreux attributs ou des attributs de grande taille. Utilisez la valeur `false` si une session a peu d'attributs, car dans ce cas tous les attributs sont stockés dans la même clé dans la grille de données.

Dans la précédente implémentation à base de filtres, il était fait référence à cette propriété en tant que mécanisme de persistance avec, comme valeurs possibles, ObjectGridStore (fragmentation) et ObjectGridAtomicSessionStore (non-fragmentation).

objectGridType

Valeur de type chaîne `REMOTE` ou `EMBEDDED`. La valeur par défaut est `REMOTE`.

Si la valeur est `REMOTE`, les données de session sont stockées en dehors du serveur sur lequel l'application Web est exécutée.

Si la valeur est `EMBEDDED`, un conteneur intégré eXtreme Scale démarre dans le processus serveur d'applications sur lequel l'application Web s'exécute.

objectGridName

Valeur de chaîne qui définit le nom de l'instance ObjectGrid utilisée pour une application Web particulière. Le nom par défaut est session.

Cette propriété doit refléter le nom objectGridName dans les fichiers XML ObjectGrid et XLM de déploiement utilisés pour démarrer les serveurs de conteneur eXtreme Scale.

objectGridXML

L'emplacement du fichier `objectgrid.xml`. Le fichier XML intégré regroupé dans la bibliothèque eXtreme Scale est chargé automatiquement si `objectGridType=EMBEDDED` et que la propriété **objectGridXML** n'est pas définie.

objectGridDeploymentXML

Indique l'emplacement du fichier XML de stratégie de déploiement d'objectGrid. Le fichier XML intégré regroupé dans la bibliothèque eXtreme Scale est chargé automatiquement si `objectGridType=EMBEDDED` et que la propriété **objectGridDeploymentXML** n'est pas définie.

replicationInterval

Entier (en secondes) qui définit le temps séparant deux écritures de sessions actualisées vers la grille. La valeur par défaut est `10` secondes. Les valeurs possibles sont comprises entre 0 et 60. 0 signifie que les sessions actualisées sont écrites dans la grille pour chaque demande dès la fin de l'appel à la méthode de service du servlet. Une valeur **replicationInterval** plus élevée améliore les performances, car un moins grand nombre de mises à jour sont écrites dans la grille de données. Mais en même temps, une valeur supérieure à 0 rend la configuration moins tolérante aux pannes.

Ce paramètre ne s'applique que lorsque objectGridType a la valeur `REMOTE`.

reuseSessionID

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. A la valeur `true` si le conteneur Web sous-jacent réutilise les ID de session dans les requêtes aux différents hôtes. La valeur de cette propriété doit être la même que la valeur du conteneur Web. Si vous utilisez WebSphere Application Server et configurez la persistance de session HTTP eXtreme Scale en utilisant la console d'administration ou le scriptage de l'outil **wsadmin**, la propriété personnalisée du conteneur Web `HttpSessionIdReuse=true` est ajoutée par défaut. **reuseSessionID** a également la valeur `true`. Si vous ne voulez pas réutiliser l'ID de session, définissez la propriété `HttpSessionIdReuse=false` dans la propriété personnalisée du conteneur Web avant de configurer la persistance de session eXtreme Scale.

sessionIdOverrideClass

Nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.xs.sessionmanager.SessionIDOverride. Cette classe est utilisée pour remplacer l'identificateur de session unique obtenu avec la méthode HttpSession.getId() afin que toutes les applications aient le même ID. Par défaut, l'ID provenant de HttpSession.getId() est utilisé.

sessionStatsSpec = session.all = enabled

Chaîne de spécification des statistiques HTTP client eXtreme Scale.

shareSessionsAcrossWebApps

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Spécifie si les sessions sont partagées entre des applications Web ; indiquez la valeur de chaîne `true` ou `false`. La spécification de servlet stipule que les sessions HTTP ne peuvent pas être partagées entre des applications Web. Une extension à la spécification de servlet est fournie pour permettre ce partage.

sessionTableSize

Entier qui définit le nombre de références de session conservées en mémoire. La valeur par défaut est `1000`.

Ce paramètre appartient uniquement à une topologie `REMOTE`, car la topologie `EMBEDDED` a déjà les données de session dans le même groupe que le conteneur Web.

Les sessions sont expulsées de la table interne en fonction de la logique LRU (least recently used). Lorsqu'une session est expulsée de cette table, elle est invalidée dans le conteneur Web. Cependant, les données ne sont pas pour autant supprimées de la grille, ce qui permet aux demandes ultérieures de cette session de continuer à extraire les données. Cette valeur doit être supérieure à la valeur maximale du pool d'unités d'exécution du conteneur Web, ce qui réduit les conflits au niveau du cache de session.

`securityEnabled`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Ce paramètre active la sécurité du client eXtreme Scale. Il doit correspondre au paramètre **`securityEnabled`** dans le fichier des propriétés sur serveur eXtreme Scale. Si les paramètres ne correspondent pas, une exception est générée.

`sessionIdOverrideClass`

Remplace l'ID session extrait d'une application. L'ID provenant de la méthode HttpSession.getId() est utilisé par défaut. Permet aux sessions HTTP client eXtreme Scale de remplacer l'ID de session unique d'une application afin que toutes les applications soient récupérées avec le même ID. Défini sur l'implémentation de l'interface com.ibm.websphere.xs.sessionmanager.SessionIDOverride. Cette interface détermine l'ID HttpSession d'après l'objet HttpServletRequest.

`traceSpec`

Spécifie la spécification de trace d'IBM® WebSphere comme une valeur de chaîne. Utilisez ce paramètre pour des serveurs d'applications autres que WebSphere Application Server.

`traceFile`

Spécifie l'emplacement du fichier de trace sous forme de valeur de chaîne. Utilisez ce paramètre pour des serveurs d'applications autres que WebSphere Application Server.

`useURLEncoding`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Affectez-lui la valeur `true` pour activer la réécriture d'URL. La valeur par défaut est `false`, ce qui indique que les cookies sont utilisés pour stocker les données de session. La valeur de ce paramètre doit être identique à celle des paramètres de conteneur Web pour la gestion des sessions.

`useCookies`

Valeur de type chaîne `true` ou `false`. A la valeur true si le conteneur Web sous-jacent réutilise les ID de session dans les requêtes aux différents hôtes. La valeur par défaut est `false`. La valeur de cette propriété doit être la même que la valeur définie dans le conteneur Web.

Rubrique parent : Création de grilles de données de session

Fichier `splicer.properties`

Le fichier `splicer.properties` contient toutes les options de configuration qui permettent de configurer un gestionnaire de sessions basé sur un filtre de servlet.

Exemple de fichier splicer.properties

Si vous décidez d'utiliser l'une des propriétés supplémentaires décrites dans ce fichier, veillez à supprimer la mise en commentaire des lignes des propriétés à activer.

```
# Fichier de propriétés qui contient toutes les options de configuration
# que le gestionnaire de sessions ObjectGrid basé sur un filtre de servlet peut être
configuré pour utiliser.
#
# Ce fichier de propriétés peut être généré pour attribuer toutes les
# valeurs par défaut à ces paramètres de configuration, et permettre de
# remplacer les paramètres individuels à l'aide des propriétés de tâche ANT,
# si ce fichier de propriétés est utilisé avec
# la tâche ANT filtersplicer.

# Valeur de chaîne "REMOTE" ou "EMBEDDED". La valeur par défaut est REMOTE.
# Si elle est définie sur "REMOTE", les données de session seront stockées en dehors du
# serveur où est exécutée l'application Web. Si sa valeur est
# "EMBEDDED", un conteneur WebSphere eXtreme Scale imbriquée démarre
# dans le processus de serveur d'applications dans lequel l'application Web est exécutée.

objectGridType = REMOTE

# Valeur de chaîne qui définit le nom de l'instance ObjectGrid
# utilisée pour une applications Web donnée. Le nom par défaut
# est session. Cette propriété doit refléter l'objectGridName dans les deux
# fichiers xml objectgrid et de déploiement utilisés pour démarrer les conteneurs eXtreme
# Scale.

objectGridName = session

# Le serveur de catalogues peut être contacté pour obtenir une instance
# ObjectGrid côté client. La valeur doit avoir le format
# "host:port<,hôte:port>", où hôte représente l'hôte d'écoute
# sur lequel le serveur de catalogue est en cours d'exécution, et port représente
# le port d'écoute du processus du serveur de catalogue. Cette liste
# peut être d'une longueur quelconque et n'est utilisée que pour l'amorçage.
# La première adresse valide est utilisée. Elle est facultative dans WebSphere
# si la propriété catalog.services.cluster est définie.

# catalogHostPort = host:port<,hôte:port>

# Entier (secondes) qui définit la durée en secondes entre
# l'écriture de sessions actualisées dans ObjectGrid. La valeur par défaut est 10. Cette
propriété
# est utilisée uniquement lorsque objectGridType a la valeur REMOTE. Les valeurs possibles
sont
# comprises entre 0 et 60. 0 signifie que les sessions actualisées sont écrites dans
l'ObjectGrid
# à la fin de l'appel à la méthode de service de servlet de chaque demande.

replicationInterval = 10

# Entier qui définit le nombre de références de session
# conservées en mémoire. La valeur par défaut est 1 000. Cette propriété n'est utilisée
# que lorsqueobjectGridType a la valeur REMOTE. Lorsque le nombre de sessions stockées
# dans la mémoire dans le conteneur Web dépasse cette valeur, la première session ayant
# fait l'objet d'un accès est invalidée depuis le conteneur Web. Si une demande
# arrive pour cette session une fois qu'elle a été invalidée, une nouvelle session
# est créée (avec un nouvel ID de session reuseSessionId=false),
# remplie avec les attributs de la session invalidée. Cette value doit toujours être
# supérieure à la taille maximale du pool d'unités
# d'exécution du conteneur pour éviter les conflits dans ce cache de session.
```

```
sessionTableSize = 1000

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "true".
# Permet de contrôler si nous stockons les données de session comme entrée
# intégrale ou de stocker chaque attribut séparément.
# Cette propriété s'appelle persistenceMechanism dans l'implémentation
# basée sur un filtre précédente, avec les valeurs possibles
# ObjectGridStore (fragmenté) et ObjectGridAtomicSessionStore
# (non fragmenté).

fragmentedSession = true

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "false".
# Active la sécurité du client eXtreme Scale. Ce paramètre doit correspondre
# au paramètre securityEnabled dans le fichier des propriétés du serveur eXtreme
# Scale. Si les paramètres ne correspondent pas, une exception est générée.

securityEnabled = false

# Spécifie la prise en charge de l'authentification des données d'identification du
client.
#   Les valeurs possibles sont les suivantes :
#   Jamais : le client ne prend pas en charge l'authentification des données
d'identification.
#   Pris en charge* : le client prend en charge l'authentification des données
d'identification si et seulement si le serveur
#                     la prend en charge également.
#   Requise : le client requiert l'authentification des données d'identification.
#   Elle est prise en charge par défaut.

# credentialAuthentication =

# Indique le nombre de tentatives d'authentification si les données d'identification
# ont expiré. Si la valeur est 0, aucune tentative d'authentification
# n'a lieu.

# authenticationRetryCount =

# Indique le nom de la classe qui implémente l'interface
# com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator.
# Cette classe est utilisée pour obtenir les données d'identification des clients.

# credentialGeneratorClass =

# Spécifie les propriétés de la classe d'implémentation
# CredentialGenerator. Les propriétés sont définies dans l'objet avec la méthode
# setProperties(String). La valeur credentialGeneratorProps n'est utilisée que si
# la valeur de la propriété credentialGeneratorClass est null.

# credentialGeneratorProps =

# Emplacement du fichier xml objectgrid.
# Le fichier xml pré-intégré qui est regroupé dans la bibliothèque eXtreme Scale
# sera automatiquement chargé si cette propriété
# n'est pas spécifiée et que objectGridType=EMBEDDED

# objectGridXML =

# Emplacement du fichier xml de stratégie de déploiement objectGrid.
# Le fichier xml pré-intégré qui est regroupé dans la bibliothèque eXtreme Scale
# sera automatiquement chargé si cette propriété
# n'est pas spécifiée et que objectGridType=EMBEDDED

# objectGridDeploymentXML =

# Chaîne de spécification de trace IBM WebShere,
# utile pour tous les autres serveurs d'applications, outre WebSphere.
```

```
# traceSpec =

# Chaîne d'emplacement de fichier de trace. # utile pour tous les autres serveurs
d'applications, outre WebSphere.

# traceFile=

# Cette propriété doit être définie pour que les sessions soient
# accessibles sur les hôtes. La valeur sera le nom du domaine
# commun aux hôtes.

# cookieDomain=

# Cette propriété doit être affectée du chemin que vous avez configuré
# pour les paramètres de cookie de serveur d'applications. Le chemin par défaut
# est /.

# cookiePath

# A la valeur true si le conteneur Web sous-jacent
# réutilise l'ID dans les demandes à différents hôtes. La valeur par défaut
# est false. La valeur doit correspondre à celle définie dans
# le conteneur Web.

# reuseSessionId=

# Une valeur de type chaîne "true" ou "false". La valeur par défaut est
# "false". Conformément à la spécification de servlet, les sessions HTTP
# ne peuvent pas être partagées dans les applications Web. Une extension à la
spécification de servlet
# est fournie pour autoriser le partage.

# shareSessionsAcrossWebApps = false

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "false". # Affectez-
lui la valeur true si vous voulez activer la réécriture d'URL (urlRewriting). La valeur
par défaut est
# false. La valeur doit être identique à celle définie dans
# les paramètres de conteneur Web de la gestion de session.

# useURLEncoding = false

# False si vous voulez désactiver les cookies en tant que
# mécanisme de suivi. La valeur par défaut est true. La valeur doit être identique à celle
définie dans
# les paramètres de conteneur Web de la gestion de session.

# useCookies = true

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "false".
# Active le suivi des statistiques des sessions HTTP client eXtreme Scale.

# enableSessionStats = false

# Remplace l'ID session extraite d'une application. # L'ID provenant de la méthode
HttpSession.getId() est utilisé par défaut.
# Permet aux sessions HTTP client eXtreme Scale de remplacer
# l'ID session unique d'une application afin que toutes les applications soient extraites
# avec le même ID.
# Défini avec l'implémentation de
# l'interface com.ibm.websphere.xs.sessionmanager.SessionIDOverride.
# Cette interface détermine l'ID HttpSession d'après
# l'objet HttpServletRequest.

# sessionIdOverrideClass =  # Spécification des statistiques HTTP client eXtreme Scale .

# sessionStatsSpec = session.all = enabled
```

```
# True si votre environnement contient plusieurs applications qui
# utilisent des noms de cookie. False, qui suppose que toutes les applications
# utilisent le même nom de cookie.

# applicationQualifiedCookies=false
```

Rubrique parent : Création de grilles de données de session

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données
Administration à l'aide de l'interface de commande HTTP
Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications

Information associée:
Installation de fichiers d'application d'entreprise à l'aide de la console
Installation d'applications d'entreprise à l'aide d'un script wsadmin

Création de grilles de données en cache dynamique

IBM® WebSphere DataPower XC10 Appliance permet de stocker des données pour une instance de cache dynamique WebSphere Application Server. En configurant cette fonction, vous permettez aux applications qui utilisent une instance de cache dynamique WebSphere Application Server d'utiliser les fonctions et les fonctionnalités de performance du dispositif.

Avant de commencer

- Vous devez installer WebSphere eXtreme Scale Client dans votre configuration WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client.
- Si la sécurité de la couche de transport du dispositif est activée ou que vous voulez que les clients utilisent la sécurité de la couche de transport, vous devez activer également la sécurité globale dans la console d'administration WebSphere Application Server. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

Pourquoi et quand exécuter cette tâche

Lorsque vous configurez le fournisseur de mémoire cache dynamique dans WebSphere Application Server pour utiliser DataPower XC10 Appliance, les données de cache sont stockées en dehors de la topologie WebSphere Application Server. Toutes les données de cache sont stockées sur le dispositif. La mémoire utilisée pour la mise en cache de vos serveurs d'applications peut être utilisée à d'autres fins.

Pour savoir comment utiliser des grilles de données de cache dynamique WebSphere eXtreme Scale Client et DataPower XC10 Appliance avec IBM WebSphere Commerce, consultez les rubriques suivantes dans la documentation d'IBM WebSphere Commerce :

- Activation du service de cache dynamique et mise en cache de servlet
- Activation du cache de données WebSphere Commerce

Procédure

1. Créer une instance de mémoire cache à configurer avec DataPower XC10 Appliance. Pour plus d'informations, voir Configuration des instances de cache dynamique..
2. Configurer le service de catalogue. Le service de catalogue permet à la configuration de la mémoire cache dynamique de WebSphere Application Server de communiquer avec DataPower XC10 Appliance. Vous pouvez configurer le service de catalogue dans la console d'administration de WebSphere Application Server en créant un domaine de service de catalogue. Pour plus d'informations, voir Création de domaines de service de catalogue dans WebSphere Application Server.
3. Créez la grille de données sur DataPower XC10 Appliance et configurez les paramètres de sécurité nécessaires. Vous pouvez exécuter le script `dynaCfgToAppliance` ou créer la configuration manuellement.

   Création de la configuration de la grille de données à l'aide du script `dynaCfgToAppliance` :

   Ce script est installé dans le répertoire `bin` du profil du gestionnaire de déploiement lorsque vous installez WebSphere eXtreme Scale Client. Avant d'exécuter ce script, vérifiez que le dispositif et le gestionnaire de déploiement sont en cours d'exécution :

   ```
   dynaCfgToAppliance <adresse_IP> <nom_jndi_cache> <admin_dispositif> <mdp_admin>
   <port_SOAP> <soap.client.props>
   ```

   `Adresse_IP`

   Spécifie l'adresse IP du système DataPower XC10 Appliance sur lequel vous souhaitez stocker vos données de mémoire cache dynamique.

   `nom_jndi_cache`

   Indique le nom du cache dynamique. Si le nom JNDI du cache dynamique contient des barres obliques (`/`), celles-ci sont converties en tirets pour le nom de la grille de données dans DataPower XC10 Appliance. Par exemple, si le nom du cache dynamique est `services/cache1`, la grille de données créée dans le dispositif s'appelle `services-cache1`. Vous ne pouvez pas utiliser les caractères suivants dans le nom de la grille de données dans DataPower XC10 Appliance : `^ . \\ / , # $ @ : ; \ * ? < > | = + & % [ ] " "`.

   `admin_dispositif`

   Spécifie l'ID administrateur à utiliser pour authentification auprès de l'interface utilisateur de DataPower XC10 Appliance.

   `mdp_admin`

   Spécifie le mot de passe administrateur à utiliser pour authentification auprès de l'interface

utilisateur de DataPower XC10 Appliance.

port_SOAP

(Facultatif) Spécifie le port SOAP du gestionnaire de déploiement si vous n'utilisez pas le port par défaut (8879).

soap.client.props

(Facultatif) Définit le chemin d'accès au fichier `soap.client.props`. Vous devez spécifier ce fichier si vous avez activé la sécurité dans WebSphere Application Server. Ce fichier active la sécurité SOAP et définit le nom d'utilisateur et le mot de passe pour administrer le gestionnaire de déploiement WebSphere Application Server :

```
com.ibm.SOAP.securityEnabled=true
com.ibm.SOAP.loginUserid=
com.ibm.SOAP.loginPassword=
```

Voir fichiers de propriétés du connecteur SOAP et du connecteur Inter-Process Communications pour plus d'informations sur le fichier `soap.client.props`.

Ce script crée la grille de données sur le dispositif. Il définit également les données d'identification par ID et mot de passe spécifiques à DataPower XC10 Appliance que vous avez spécifiées avec les paramètres **admin_dispositif** et **mdp_admin** à l'aide des propriétés personnalisées suivantes :

- xc10.<*nom_grille_données*>.userid
- xc10.<*nom_grille_données*>.password

Ces noms de propriétés ne respectent pas la casse des caractères. La valeur du mot de passe est codée. Si vous exécutez de nouveau le script après la configuration initiale, les propriétés personnalisées sont mises à jour.

Création manuelle de la configuration de la grille de données :

a. Créez la grille de données de mémoire cache dynamique dans l'interface graphique de DataPower XC10 Appliance. Cliquez sur **Grille de données** > **Cache dynamique**. Le nom du cache doit correspondre au nom JNDI du cache dynamique dans la configuration WebSphere Application Server. Lorsque vous entrez le nom JNDI, remplacez les barres obliques (`/`) par des tirets pour le nom de la grille de données dans DataPower XC10 Appliance. Par exemple, si le nom du cache dynamique est `services/cache1`, la grille de données créée dans le dispositif doit s'appeler `services-cache1`.
b. Créez les propriétés personnalisées xc10.<*nom_grille_données*>.userid et xc10.<*nom_grille_données*>.password sur la cellule WebSphere Application Server. La valeur de <*nom_grille_données*> dans chaque propriété personnalisée correspond au nom JNDI de la grille de données, les barres obliques (`/`) étant remplacées par des tirets. Par exemple, dans l'exemple précédent, les noms de propriété personnalisée sont `xc10.services-cache1.userid` et `xc10.services-cache1.password`. Les valeurs doivent correspondre à un ID utilisateur et un mot de passe qui peuvent accéder à la grille de données dans la configuration de DataPower XC10 Appliance. Vous pouvez coder le mot de passe à l'aide du script `encodePassword`, qui se trouve dans le répertoire `bin` du gestionnaire de déploiement.

4. 2.5+ Facultatif : Vous pouvez choisir d'activer la réplication multimaître dans le fournisseur de cache dynamique DataPower XC10 Appliance. Pour plus d'informations, voir Configuration d'une réplication multimaître entre les collectivités.

   Remarque : Les utilisateurs de la grille du cache dynamique de WebSphere Portal Server ou de WebSphere Commerce Server peuvent avoir défini plusieurs instances de cache au sein de leur configuration WebSphere Application Server. Si vous décidez d'activer la réplication multimaître pour DataPower XC10 Appliance, cette configuration n'affecte que les instances de cache qui sont définies pour l'utilisation de DataPower XC10 Appliance en tant que fournisseur de mémoire cache dynamique ; elle n'aura aucun effet sur les instances de cache définies pour l'utilisation du fournisseur de mémoire cache dynamique WebSphere Application Server par défaut.

Résultats

La configuration du fournisseur de mémoire cache dynamique avec le dispositif permet de réduire la quantité de mémoire nécessaire aux serveurs d'applications. Toutes les données de mémoire cache sont transférées vers le dispositif et disparaisse de la mémoire des serveurs d'applications.

Que faire ensuite

- Configurez la sécurité avant de commencer à envoyer des données à la grille de données. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
- Configurez des répliques. Les répliques permettent de s'assurer que les données de vos grilles de données sont disponibles si la copie principale échoue. Pour configurer des répliques, cliquez sur

Grille de données > **Mémoire cache dynamique** > **Afficher les attributs avancés**. Les répliques ne sont créées que si le dispositif fait partie d'une collectivité. Si le nombre de dispositifs de la collectivité est *n*, le nombre maximal de répliques est *n - 1*. Ainsi, si vous configurez trois répliques mais que vous n'avez que deux dispositifs dans la collectivité, une seule réplique est créée. Des répliques supplémentaires sont créées si vous ajoutez des dispositifs à la collectivité. Indiquez comme nombre de répliques le nombre idéal de répliques souhaitées : ainsi, lorsque des dispositifs rejoignent la collectivité, de nouvelles répliques sont créées. Le contenu de la grille de données est effacé lorsque vous modifiez le nombre de répliques.

- La limite de capacité de la grille pour une instance de mémoire cache dynamique WebSphere Application Server est configurée lors de la création de cette instance. Si cette taille est dépassée, une stratégie d'éviction LRU (least recently used) est utilisée pour maintenir le nombre d'entrées dans la limite configurée.
- Vous pouvez contrôler la grille de données de mémoire cache dynamique à partir de l'interface utilisateur de DataPower XC10 Appliance. Pour plus d'informations, voir Surveillance des grilles de données dans l'interface utilisateur.

Java 2.5+ **Configuration des instances de cache dynamique**

Le service de mise en mémoire cache de WebSphere Application Server permet la création d'une instance de cache par défaut (baseCache) et d'instances de cache de servlet et d'objet supplémentaires.

Java 2.5+ **Présentation du fournisseur de cache dynamique**

WebSphere Application Server fournit un service de cache dynamique disponible pour déployer des applications Java EE. Ce service permet de mettre des données en cache, telles que la sortie d'un servlet, une page JSP ou des commandes, ainsi que les données d'objet définies par programme dans une application d'entreprise en utilisant des API DistributedMap. En configurant cette fonction, vous pouvez activer les applications qui utilisent le service de cache dynamique pour qu'elles utilisent les fonctions et les fonctionnalités de performance de WebSphere DataPower XC10 Appliance.

Java **Création de domaines de service de catalogue dans WebSphere Application Server**

A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données de cache dynamique.

Java 2.5+ **Configuration d'un cache local pour la mémoire cache dynamique**

Vous pouvez configurer un cache local afin d'utiliser une grille de données de mémoire cache dynamique sur le dispositif ou la collectivité. Le cache local utilise des ressources JVM locales. Généralement, le cache local comporte un sous-ensemble des données qui figurent dans la grille de données de cache dynamique sur le dispositif.

Rubrique parent : Configuration des grilles de données

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Configuration des instances de cache dynamique

Le service de mise en mémoire cache de WebSphere Application Server permet la création d'une instance de cache par défaut (baseCache) et d'instances de cache de servlet et d'objet supplémentaires.

Pourquoi et quand exécuter cette tâche

L'instance de cache par défaut (baseCache) était initialement la seule instance de cache dynamique prise en charge par WebSphere Application Server et elle est actuellement l'instance de cache dynamique standard utilisée par WebSphere Commerce Suite. Les instances supplémentaires de servlet et d'objet ont été ajoutées dans les dernières versions de WebSphere Application Server et sont configurées dans une section **Instance de Cache** distincte de la console d'administration WebSphere Application Server.

Java 2.5+ **Configuration de l'instance de mémoire cache dynamique par défaut (baseCache)**
L'instance de mémoire cache dynamique par défaut est créée par le service de mise en mémoire cache dynamique WebSphere Application Server. Cette instance de mémoire cache dynamique de servlet est utilisée par des produits tels qu'IBM WebSphere Commerce. Contrairement aux autres instances de mémoire cache définies avec WebSphere Application Server, baseCache est propre à un seul serveur ou à une seule instance de cluster. Utilisez cette procédure pour configurer l'instance baseCache dans WebSphere Application Server afin de l'utiliser comme fournisseur de cache dynamique avec WebSphere eXtreme Scale.

Java 2.5+ **Configuration des instances de cache dynamique d'objet ou de servlet**
WebSphere Application Server permet de configurer des instances de cache dynamique d'objet ou de servlet en plus de l'instance par défaut. Pour configurer ces instances supplémentaires, procédez comme suit.

Java 2.5+ **Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées**
WebSphere Application Server permet de définir des propriétés personnalisées dans une instance de cache dynamique.

Rubrique parent : Java Création de grilles de données en cache dynamique

Java

Configuration de l'instance de mémoire cache dynamique par défaut (baseCache)

L'instance de mémoire cache dynamique par défaut est créée par le service de mise en mémoire cache dynamique WebSphere Application Server. Cette instance de mémoire cache dynamique de servlet est utilisée par des produits tels qu'IBM WebSphere Commerce. Contrairement aux autres instances de mémoire cache définies avec WebSphere Application Server, baseCache est propre à un seul serveur ou à une seule instance de cluster. Utilisez cette procédure pour configurer l'instance baseCache dans WebSphere Application Server afin de l'utiliser comme fournisseur de cache dynamique avec WebSphere eXtreme Scale.

Avant de commencer

- Pour que vous puissiez utiliser le fournisseur de cache dynamique, WebSphere eXtreme Scale Client doit être installé sur les déploiements de noeud WebSphere Application Server, et notamment le noeud du gestionnaire de déploiement. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server .
- Le domaine de service de catalogue WebSphere eXtreme Scale doit être configuré. Pour plus d'informations, voir Création de domaines de service de catalogue dans WebSphere Application Server.
-
- Si les serveurs de catalogue dans le domaine de service de catalogue utilisent SSL (Secure Sockets Layer) ou si vous voulez utiliser SSL pour un domaine de service de catalogue qui prend en charge SSL, vous devez activer la sécurité globale dans la console d'administration de WebSphere Application Server. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Cette procédure s'applique à la version 8.0 de la console d'administration de WebSphere Application Server. Ces informations peuvent varier en fonction la version de WebSphere Application Server que vous utilisez.

Remarque :

- WebSphere eXtreme Scale version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.
- La procédure suivante s'applique à la topologie de mémoire cache dynamique WebSphere eXtreme Scale distante. Toutes les autres topologies, notamment intégrées, partitionnées intégrées et locales, sont obsolètes dans WebSphere eXtreme Scale version 8.6.
- La procédure ci-dessous suppose que vous utilisez WebSphere Application Server version 7.0 groupe de correctifs 27, version 8.0 groupe de correctifs 6, version 8.5 groupe de correctifs 2 ou ultérieur. L'APAR PM71992 WebSphere Application Server est inclus dans ces versions.

Procédure

1. Démarrez la console d'administration WebSphere Application Server.
2. Dans le menu supérieur, cliquez sur **Serveurs > Type de serveur > Serveurs d'applications WebSphere**.
3. Dans la zone **Serveurs d'applications**, sélectionnez le **nom de votre serveur**.
4. Dans le panneau **Configuration**, cliquez sur **Services de conteneur** et sélectionnez **Service de cache dynamique**.
5. Dans la liste déroulante **Fournisseur de cache**, sélectionnez WebSphere eXtreme Scale.
6. Si vous voulez changer la taille du cache, définissez-la dans la zone de **taille de cache**. La taille de cache définit le nombre maximal d'entrées autorisées dans chaque partition dans une grille WebSphere eXtreme Scale pour l'instance de mémoire cache dynamique. La valeur par défaut est de 2000 entrées dans chaque partition.
7. Sélectionnez **Activer la réplication de cache**. Dans ce cas, les données en mémoire cache sont stockées à distance dans la grille de données et non pas localement. Lorsque vous utilisez WebSphere eXtreme Scale comme fournisseur de cache, vous devez sélectionner cette option.
8. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
9. Dans le menu supérieur, cliquez sur **Serveurs > Type de serveur > Serveurs d'applications WebSphere**.
10. Dans la zone **Serveurs d'applications**, sélectionnez le **nom de votre serveur**.
11. Dans le panneau **Configuration**, cliquez sur **Paramètres de conteneur Web** et sélectionnez **Conteneur Web**.

12. Cochez l'option d'**activation de la mise en cache de servlet**.
13. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.

Que faire ensuite

Par défaut, chaque instance de mémoire cache dynamique configurée sur WebSphere Application Server correspond à une grille de données de cache dynamique dont le nom correspond au nom JNDI de l'instance de mémoire cache. En outre, par défaut, les données de cette instance de mémoire cache sont stockées dans une mappe dynamique au sein de cette grille, et le suffixe du nom de cette mappe dynamique correspond également au nom JNDI de l'instance de mémoire cache. Par exemple, si vous configurez sur WebSphere Application Server une instance de mémoire cache portant le nom JNDI cache1, une grille de données de cache dynamique est créée sur le dispositif avec le nom cache1. Dans cette grille de données cache1, une mappe dynamique baptisée IBM_DC_PARTIONED_cache1 est créée pour stocker les données.

Dans la plupart des cas, il n'est pas nécessaire de modifier cette configuration. Toutefois, dans certaines circonstances, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant des noms JNDI différents, mappent vers différentes mappes dynamiques au sein de la même instance de grille de données. Dans d'autres cas, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant le même nom JNDI, mappent vers différentes instances de grille de données de cache dynamique ou vers différentes instances de mappe dynamique au sein de la même grille de données de cache dynamique. Par exemple, si vous avez une application qui utilise l'instance de mémoire cache dynamique par défaut (baseCache), vous pouvez vouloir utiliser le même dispositif pour vos environnements de test et de production, tout en gardant les données mises en cache dans des grilles de données distinctes ou dans des mappes dynamiques distinctes au sein de la même grille de données.

Pour contrôler cette configuration, vous pouvez définir les propriétés personnalisées suivantes pour les instances de mémoire cache :

Conseil : Ces propriétés peuvent être particulièrement utiles lorsque vous utilisez l'instance de mémoire cache dynamique par défaut (baseCache), car le nom JNDI baseCache est automatiquement attribué au cache et ne peut être modifié.

com.ibm.websphere.xs.dynacache.grid_name

Utilisez cette propriété personnalisée pour indiquer le nom de l'instance de grille de données de cache dynamique à laquelle une instance de mémoire cache dynamique correspond.

com.ibm.websphere.xs.dynacache.cache_name

Utilisez cette propriété personnalisée pour indiquer le nom à utiliser à la place du nom JNDI comme nom de la grille de données de cache dynamique et comme nom de la mappe dynamique au sein de cette grille. Si la propriété **`com.ibm.websphere.xs.dynacache.grid_name`** est également définie, cette propriété ne s'applique qu'au nom de la mappe dynamique.

Pour savoir comment définir ces propriétés, voir Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées.

Rubrique parent : Java 2.5+ Configuration des instances de cache dynamique

Java

Configuration des instances de cache dynamique d'objet ou de servlet

WebSphere Application Server permet de configurer des instances de cache dynamique d'objet ou de servlet en plus de l'instance par défaut. Pour configurer ces instances supplémentaires, procédez comme suit.

Avant de commencer

- Pour pouvoir utiliser le fournisseur de cache dynamique, WebSphere eXtreme Scale doit être installé sur les déploiements de noeud WebSphere Application Server et notamment le noeud du gestionnaire de déploiement. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server .
- Un domaine de service de catalogue WebSphere eXtreme Scale doit être configuré. Voir Création de domaines de service de catalogue dans WebSphere Application Server.
- Si les serveurs de catalogue dans le domaine de service de catalogue utilisent SSL (Secure Sockets Layer (SSL) ou que vous voulez utiliser SSL pour un domaine de service de catalogue qui le prend en charge, vous devez activer la sécurité globale dans la console d'administration WebSphere Application Server. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Dans cette procédure, vous pouvez créer deux types d'instance de cache : instance de cache d'objet et instance de cache de servlet. Une instance de cache d'objet est un emplacement en complément du cache dynamique partagé par défaut où les applications Java 2 Platform, Enterprise Edition (J2EE) peuvent stocker, distribuer et partager des objets. Après avoir configuré des instances de cache d'objet, vous pouvez utiliser l'interface DistributedMap ou DistributedObjectCache dans le package com.ibm.websphere.cache pour accéder par programme aux instances de cache. Voir API (Application Programming Interface) supplémentaires pour plus d'informations sur les interfaces DistributedMap et DistributedObjectCache . Les instances de cache de servlet sont des emplacements qui, en plus du cache dynamique par défaut, permettent au cache dynamique de stocker, distribuer et partager le résultat et les effets secondaires d'un servlet appelé. En configurant une instance de cache de servlet, vous conférez aux applications une souplesse plus grande et les ressources en mémoire cache sont mieux optimisées. Le nom JNDI (Java Naming and Directory Interface) défini pour l'instance de cache dans la console d'administration est associé à l'élément d'instance de cache dans le fichier de configuration `cachespec.xml`. Tous les éléments <cache-entry> spécifiés dans un élément <cache-instance> sont créés dans cette instance de cache spécifique. Tous les éléments <cache-entry> spécifiés en dehors d'un élément <cache-instance> sont stockés dans l'instance de cache dynamique par défaut. Voir Instances de cache pour plus d'informations sur les instances de cache de type objet et servlet.

Cette procédure s'applique à la version 8.0 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction la version WebSphere Application Server que vous utilisez.

Remarque :

- WebSphere eXtreme Scale version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Procédure

- Pour configurer un cache d'objet ou de servlet avec la console d'administration WebSphere Application Server, procédez comme suit :

  1. Démarrez la console d'administration WebSphere Application Server.
  2. Dans le menu supérieur, cliquez sur **Ressources > Instances de cache > Instances de cache d'objet**.
  3. Dans la zone des **instances de cache d'objet**, sélectionnez le type d'instance de cache à créer. Il peut s'agir d'une instance de cache d'objet ou de servlet.
  4. Définissez la portée de l'instance de cache. Spécifiez une portée de cellule pour que l'instance de cache soit disponible pour tous les serveurs de la cellule. Une portée de noeud rend l'instance de cache disponible pour tous les serveurs d'un noeud. Une portée de serveur rend l'instance de cache disponible pour le serveur sélectionné uniquement. Si nécessaire, vous pouvez combiner les portées.
  5. Cliquez sur **Appliquer** et enregistrez la portée.
  6. Cliquez sur **Nouveau** et définissez une instance de cache d'objet.
  7. Dans la liste déroulante **Fournisseur de cache**, sélectionnez WebSphere eXtreme Scale.

     Remarque : Si WebSphere eXtreme Scale n'apparaît pas comme fournisseur de cache

dynamique, cela implique que le profil WebSphere Application Server n'a pas été étendu pour WebSphere eXtreme Scale.

8. Spécifiez le nom JNDI de l'instance de cache dynamique. Pour les objets cache, ce nom sera utilisé lors de la consultation du cache. Pour les caches de servlet, il s'agit du nom d'attribut défini dans l'élément <cache-instance> dans le fichier `cachespec.xml`.
9. Spécifiez le nom JNDI de l'instance de cache dynamique.
10. Si vous voulez changer la taille du cache, redéfinissez-la dans la zone de **taille de cache**. La taille de cache définit le nombre maximal d'entrées autorisées dans chaque partition dans une grille WebSphere eXtreme Scale pour l'instance de cache dynamique. La valeur par défaut est 2000 entrées dans chaque partition.
11. Cochez la case d'**activation de la réplication de cache**. Dans ce cas, les données en cache sont stockées à distance dans la grille et non pas localement. Vous devez cocher cette case lorsque vous utilisez WebSphere eXtreme Scale comme fournisseur de cache.
12. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.

- Pour configurer un cache d'objet ou de servlet en utilisant le fichier `cacheinstances.properties`, procédez comme suit :

  1. Créez un fichier `cacheinstances.properties`. Voir Fichier de propriétés des instances de cache pour le contenu nécessaire.
  2. Placez le fichier `cacheinstances.properties` dans le serveur d'applications ou le chemin d'accès aux applications. Par exemple, vous pouvez utiliser votre fichier WAR (Web application archive) d'application, le répertoire `WEB-INF\classes` ou créer un répertoire `server_root\classes` pour l'y placer.

Que faire ensuite

Par défaut, chaque instance de mémoire cache dynamique configurée sur WebSphere Application Server correspond à une grille de données de cache dynamique dont le nom correspond au nom JNDI de l'instance de mémoire cache. En outre, par défaut, les données de cette instance de mémoire cache sont stockées dans une mappe dynamique au sein de cette grille, et le suffixe du nom de cette mappe dynamique correspond également au nom JNDI de l'instance de mémoire cache. Par exemple, si vous configurez sur WebSphere Application Server une instance de mémoire cache portant le nom JNDI cache1, une grille de données de cache dynamique est créée sur le dispositif avec le nom cache1. Dans cette grille de données cache1, une mappe dynamique baptisée IBM_DC_PARTIONED_cache1 est créée pour stocker les données.

Dans la plupart des cas, il n'est pas nécessaire de modifier cette configuration. Toutefois, dans certaines circonstances, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant des noms JNDI différents, mappent vers différentes mappes dynamiques au sein de la même instance de grille de données. Dans d'autres cas, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant le même nom JNDI, mappent vers différentes instances de grille de données de cache dynamique ou vers différentes instances de mappe dynamique au sein de la même grille de données de cache dynamique. Par exemple, si vous avez une application qui utilise l'instance de mémoire cache dynamique par défaut (baseCache), vous pouvez vouloir utiliser le même dispositif pour vos environnements de test et de production, tout en gardant les données mises en cache dans des grilles de données distinctes ou dans des mappes dynamiques distinctes au sein de la même grille de données.

Pour contrôler cette configuration, vous pouvez définir les propriétés personnalisées suivantes pour les instances de mémoire cache :

Conseil : Ces propriétés peuvent être particulièrement utiles lorsque vous utilisez l'instance de mémoire cache dynamique par défaut (baseCache), car le nom JNDI baseCache est automatiquement attribué au cache et ne peut être modifié.

com.ibm.websphere.xs.dynacache.grid_name

Utilisez cette propriété personnalisée pour indiquer le nom de l'instance de grille de données de cache dynamique à laquelle une instance de mémoire cache dynamique correspond.

com.ibm.websphere.xs.dynacache.cache_name

Utilisez cette propriété personnalisée pour indiquer le nom à utiliser à la place du nom JNDI comme nom de la grille de données de cache dynamique et comme nom de la mappe dynamique au sein de cette grille. Si la propriété **`com.ibm.websphere.xs.dynacache.grid_name`** est également définie, cette propriété ne s'applique qu'au nom de la mappe dynamique.

Pour savoir comment définir ces propriétés, voir Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées.

Rubrique parent : Java 2.5+ Configuration des instances de cache dynamique

Java

Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées

WebSphere Application Server permet de définir des propriétés personnalisées dans une instance de cache dynamique.

Avant de commencer

Vous devez avoir configuré une instance de cache par défaut ou disposer d'un objet supplémentaire ou d'une instance de cache de type servlet. Voir Configuration de l'instance de mémoire cache dynamique par défaut (baseCache) or Configuration des instances de cache dynamique d'objet ou de servlet.

Pourquoi et quand exécuter cette tâche

Vous pouvez définir des propriétés personnalisées spécifiques d'une instance de cache dynamique en procédant comme suit :

- Utilisez la console d'administration WebSphere Application Server avec l'APAR PM71992 si vous voulez définir des propriétés spécifiques d'une instance de cache dynamique donnée. Si vous ne disposez pas de ce correctif, contactez WebSphere Application Server sur la page du support http://www.ibm.com/software/webservers/appserv/was/support.
- Si vous avez créé un fichier `cacheinstances.properties`, vous pouvez définir des propriétés personnalisées dans ce fichier. Cette méthode ne peut pas être utilisée pour définir des propriétés personnalisées pour une instance de cache dynamique (baseCache) par défaut.
- Utilisez la console d'administration WebSphere Application Server pour changer les valeurs des propriétés personnalisées JVM (Java virtual machine).

  Remarque : Les propriétés JVM peuvent affecter toutes les instances de cache dans une machine JVM donnée.

  Remarque : La portée des propriétés JVM peut s'appliquer à toutes les instances de cache dans une machine JVM WebSphere Application Server JVM. Par conséquent, il est préférable d'utiliser des propriétés personnalisées de cache dans la console d'administration WebSphere Application Server (avec l'APAR PM71992 pour une instance de cache par défaut), ou un fichier `cacheinstances.properties` dans la plupart des cas.

Procédure

- Pour définir une propriété personnalisée dans une instance de cache dans la console d'administration WebSphere Application Server, procédez comme suit :
  1. Démarrez la console d'administration WebSphere Application Server.

     Remarque : Ces étapes ne peuvent pas être exécutées pour une instance (baseCache) par défaut tant que vous n'avez pas appliqué l'APAR R PM71992 WebSphere Application Server. Ce correctif est disponible dans les versions WebSphere Application Server 7.0.0.27, 8.0.0.6, 8.5.0.2 et les versions suivantes. Si vous ne disposez pas de correctif, consultez la page WebSphere Application Server Support, http://www.ibm.com/software/webservers/appserv/was/support.
  2. Accédez à l'instance de cache désirée que vous avez configurée.
  3. Dans le panneau d'instance de cache, cliquez sur **Propriétés supplémentaires > Propriétés personnalisées**.
  4. Sélectionnez **Nouveau** et définissez le nom et la valeur de la propriété personnalisée.
  5. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
  6. Redémarrez le gestionnaire de déploiement et tous les processus serveur d'applications.
- Pour définir une propriété personnalisée pour une instance de cache en utilisant le fichier `cacheinstances.properties`, procédez comme suit :

  Remarque : Ces étapes ne peuvent pas être exécutées pour une instance (baseCache) par défaut.

  1. Ajoutez la propriété personnalisée à un fichier `cacheinstances.properties`. Si vous devez créer ce fichier, voir Fichier de propriétés des instances de cache pour connaître le contenu requis.
  2. Placez le fichier `cacheinstances.properties` dans le serveur d'applications ou le chemin d'accès aux applications. Par exemple, vous pouvez utiliser votre fichier WAR (Web application archive) d'application, le répertoire `WEB-INF\classes` ou créer un répertoire `server_root\classes` pour l'y placer.
- Utilisez la console d'administration WebSphere Application Server pour changer les valeurs des

propriétés personnalisées JVM (Java virtual machine). Voir Propriétés personnalisées JVM (java virtual machine) pour plus d'informations.

Java 2.5+ **Fichier de propriétés des instances de cache**

Pour configurer un cache d'objet ou de servlet en utilisant le fichier `cacheinstances.properties`.

Java 2.5+ **Propriétés personnalisées de cache dynamique**

Reportez-vous à ce tableau pour définir les propriétés personnalisées d'une instance de cache dynamique par défaut ou d'une instance de cache de servlet ou d'objet.

Rubrique parent : Java 2.5+ Configuration des instances de cache dynamique

Fichier de propriétés des instances de cache

Pour configurer un cache d'objet ou de servlet en utilisant le fichier `cacheinstances.properties`.

Tableau 1. Propriétés des instances de cache

Nom de la propriété : – x est le numéro de l'instance	Obligatoire	Portée	Valeur possible	Description
cache.instance.x	Oui	Par instance de cache	n'importe quelle chaîne (pas de définition par défaut)	Spécifie le nom de l'instance du cache ou le nom JNDI.
cache.instance.x.cacheSize	Non	Par instance de cache	> 0 (par défaut =2000)	Indique le nombre maximal d'entrées autorisées dans une partition dans la grille WebSphere eXtreme Scale. Multipliez cette valeur par le nombre de partitions pour déterminer la capacité du cache dans la grille WebSphere eXtreme Scale.
cache.instance.x.createCacheAtServerStartup	Non	Par instance de cache	True ou false (par défaut=false)	Indique si l'instance de cache configurée est créée lors du démarrage du serveur.
cache.instance.x.enableServletSupport	Non	Par instance de cache	True ou false (par défaut=false)	Spécifie si l'instance de cache correspond au cache des servlets ou au cache d'objets.
cache.instance.x.enableCacheReplication	Oui (uniquement jusqu'à l'APAR)	Par instance de cache	True ou false (par défaut=false)	Indique que le cache est éloigné du serveur d'applications. Cette propriété doit avoir la valeur True dans la topologie distante WebSphere eXtreme Scale.
cache.instance.x.cacheProviderName	Oui	Par instance de cache	com.ibm.ws.objectgrid.dynacache.CacheProviderImpl	Indique le fournisseur de cache dynamique. Le fournisseur WebSphere Application Server est utilisé par défaut si WebSphere eXtreme Scale n'est pas défini.
cache.instance.x.ignoreValueInInvalidationEvent	Non	Par instance de cache	True ou false (par défaut=false)	Spécifie si la valeur de cache de l'événement d'invalidation est

		cache		à invalidation est ignorée. Si la propriété est définie comme true, la valeur en cache de l'événement est défini comme NULL lorsque le code est retourné à l'appelant.
cache.instance.x.<custom property>	Dépend de la propriété à ajouter.	Par instance de cache	Dépend de la propriété à ajouter.	Vous pouvez ajouter n'importe quelle propriété à ce fichier.

Rubrique parent : Java 2.5+ Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées

Propriétés personnalisées de cache dynamique

Reportez-vous à ce tableau pour définir les propriétés personnalisées d'une instance de cache dynamique par défaut ou d'une instance de cache de servlet ou d'objet.

Tableau 1. Propriétés personnalisées de cache dynamique

Nom de la propriété	Obligatoire	Portée	Valeur possible	Description
com.ibm.websphere.xs.dynacache.topology	Oui (après l'application de l'APAR PM71992 de WebSphere Application Server, cette propriété n'est plus nécessaire.)	Par instance de cache	`remote`	Indique la topologie de l'instance de cache. La topologie `remote` est la seule topologie utilisable avec WebSphere DataPower XC10 Appliance.
com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent	Non	Par instance de cache	`true` ou `false` Valeur par défaut : `true`	Spécifie si la valeur en cache de l'événement d'invalidation est ignorée. Si la propriété est définie comme true, la valeur en cache de l'événement est défini comme NULL lorsque le code est retourné à l'appelant.
com.ibm.websphere.xs.dynacache.ignore_value_in_change_event	Non	Par instance de cache	`true` ou `false` Valeur par défaut : `true`	Spécifie si la valeur en cache de l'événement de changement est ignorée. Si la valeur est true, la valeur de cache de l'événement de changement est NULL lorsque le code est retourné à l'appelant.
com.ibm.websphere.xs.dynacache.cs_override	Non	Par instance de cache	Noeud final de service de catalogue Exemple : `9.5.12.345:2819`	Indique le noeud final de service de catalogue de la grille de données à associer à cette instance de cache. Cette zone est obligatoire si elle n'est pas spécifiée dans la console d'administration

				d'administration WebSphere Application Server.
com.ibm.websphere.xs.dynacache.grid_name	Non	Par instance de cache	N'importe quelle chaîne Valeur par défaut : `nom JNDI de l'instance de cache`	Indique le nom de la grille de données que vous avez créée.
com.ibm.websphere.xs.dynacache.map_template_name	Non	Par instance de cache	Vous pouvez utiliser l'un des modèles de mappe suivants : • `IBM_DC_PARTITIONED_.*` (valeur par défaut) • `IBM_DC_NCI_PARTITIONED_.*` (Indique que ce cache utilise un cache local.)	Indique le nom du préfixe de mappe de modèle.
com.ibm.websphere.xs.dynacache.cache_name	Non	Par instance de cache	N'importe quelle chaîne Valeur par défaut : `valeur de cache.instance.x`	Définit le nom du suffixe unique qui est utilisé comme nom de la mappe de modèle. Par exemple, `IBM_DC_PARTITIONED.<nom_cache>`
com.ibm.websphere.xs.dynacache.near_cache_size	Non	Par instance de cache	Valeur supérieur à zéro. Valeur par défaut : `valeur spécifiée dans cache.instance.x.cacheSize`	Indique le nombre maximal d'entrées autorisées dans une instance de cache local. Par défaut, cette valeur est égale au nombre maximal d'entrées autorisées dans une partition dans l'ObjectGrid distant de cette instance de cache.
com.ibm.websphere.xs.dynacache.request_retry_timeout_override	Non	Par instance de cache	Valeur entière représentant des millisecondes : • `-1` : Infini. Les requêtes n'expirent jamais. • `0` : Les requêtes expirent	Définit le temps (en millisecondes) pendant lequel une requête peut s'exécuter avant qu'un dépassement de délai ne se produise. Cette propriété remplace la propriété **`requestRetryTimeout`** si elle est définie dans le fichier de propriétés

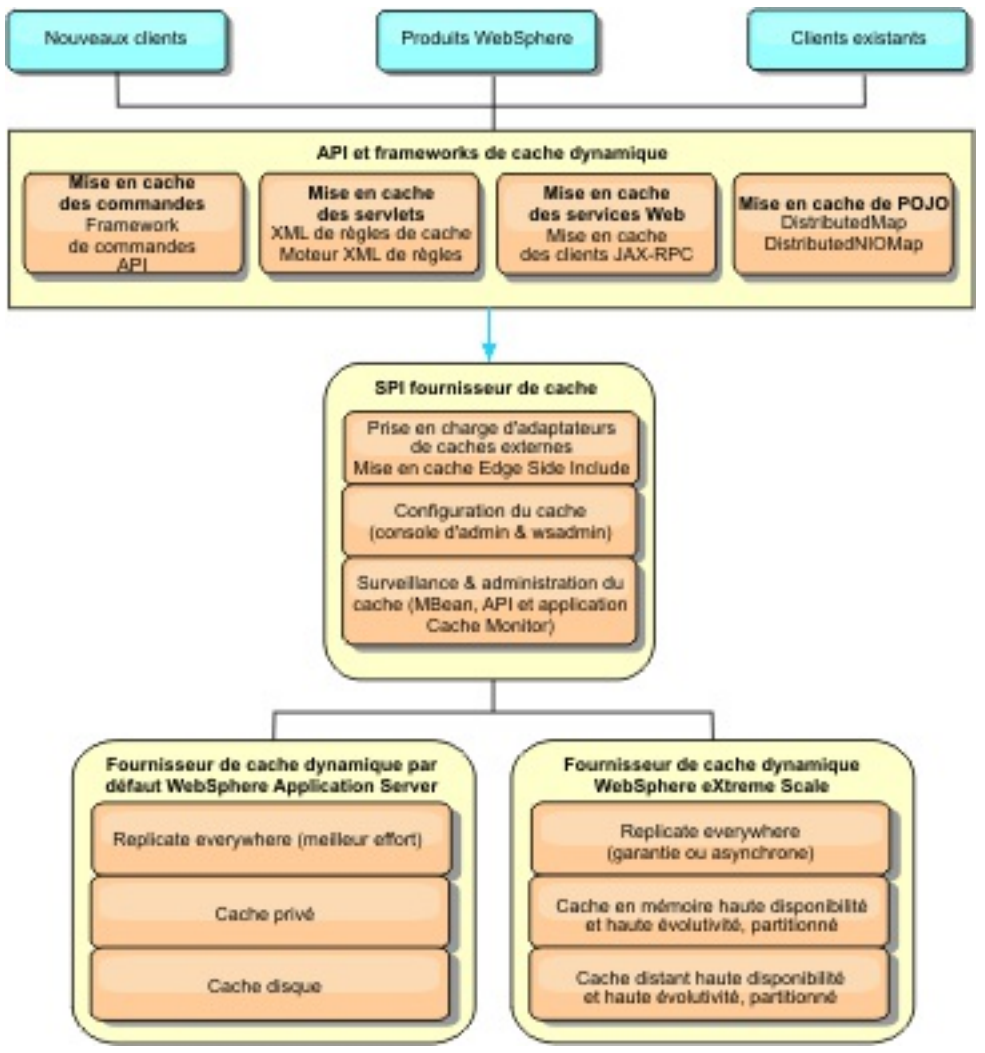
				immédiatement. • >0 : Délai (en millisecondes) pendant lequel les requêtes peuvent s'exécuter avant d'expirer. • 2000 : Valeur par défaut adoptée si vous n'utilisez pas cette propriété personnalisée ou le fichier de propriétés du client pour définir cette valeur.	du client. Pour plus d'informations, voir Fichier de propriétés du client. Le délai d'attente par défaut pour les nouvelles tentatives d'exécution des requêtes pour les instances de cache dynamique est de 2000 millisecondes s'il n'est pas défini à l'aide de cette propriété personnalisée ou dans le fichier des propriétés du client.

Rubrique parent : Java 2.5+ Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées

Présentation du fournisseur de cache dynamique

Java WebSphere Application Server fournit un service de cache dynamique disponible pour déployer des applications Java™ EE. Ce service permet de mettre des données en cache, telles que la sortie d'un servlet, une page JSP ou des commandes, ainsi que les données d'objet définies par programme dans une application d'entreprise en utilisant des API DistributedMap. En configurant cette fonction, vous pouvez activer les applications qui utilisent le service de cache dynamique pour qu'elles utilisent les fonctions et les fonctionnalités de performance de WebSphere DataPower XC10 Appliance.

Initialement, le seul fournisseur de service pour le service de cache dynamique était le moteur de cache dynamique par défaut intégré dans WebSphere Application Server. Vous pouvez également définir WebSphere DataPower XC10 Appliance comme fournisseur de cache pour une instance de cache. En configurant cette fonction, vous pouvez activer des applications qui utilisent le service de mise en mémoire cache dynamique, pour utiliser les fonctions et les fonctions de performances de .



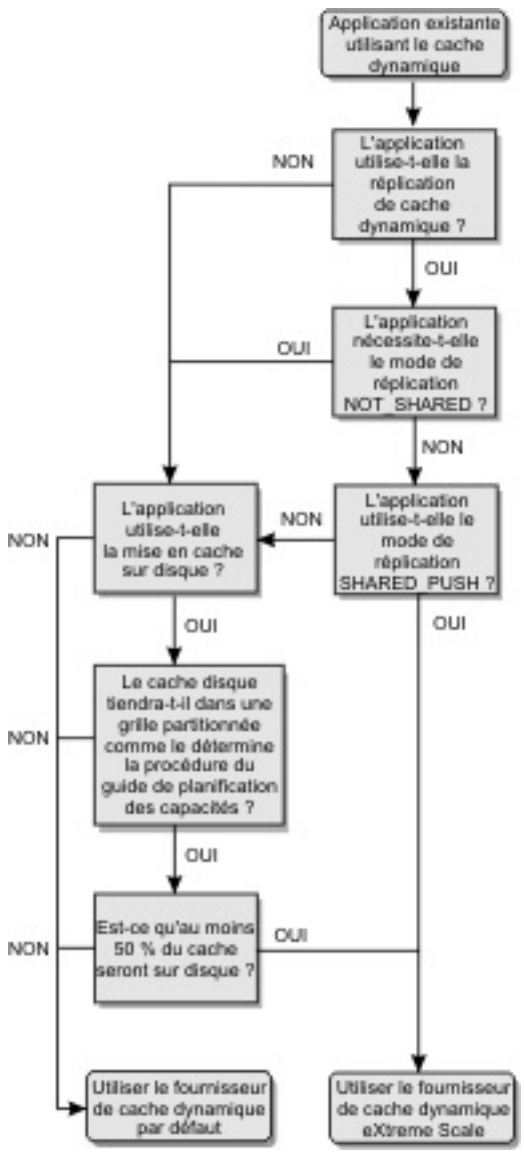
Vous pouvez installer et configurer le fournisseur de cache dynamique comme décrit dans Configuration de l'instance de mémoire cache dynamique par défaut (baseCache).

Choix du mode d'utilisation de WebSphere DataPower XC10 Appliance

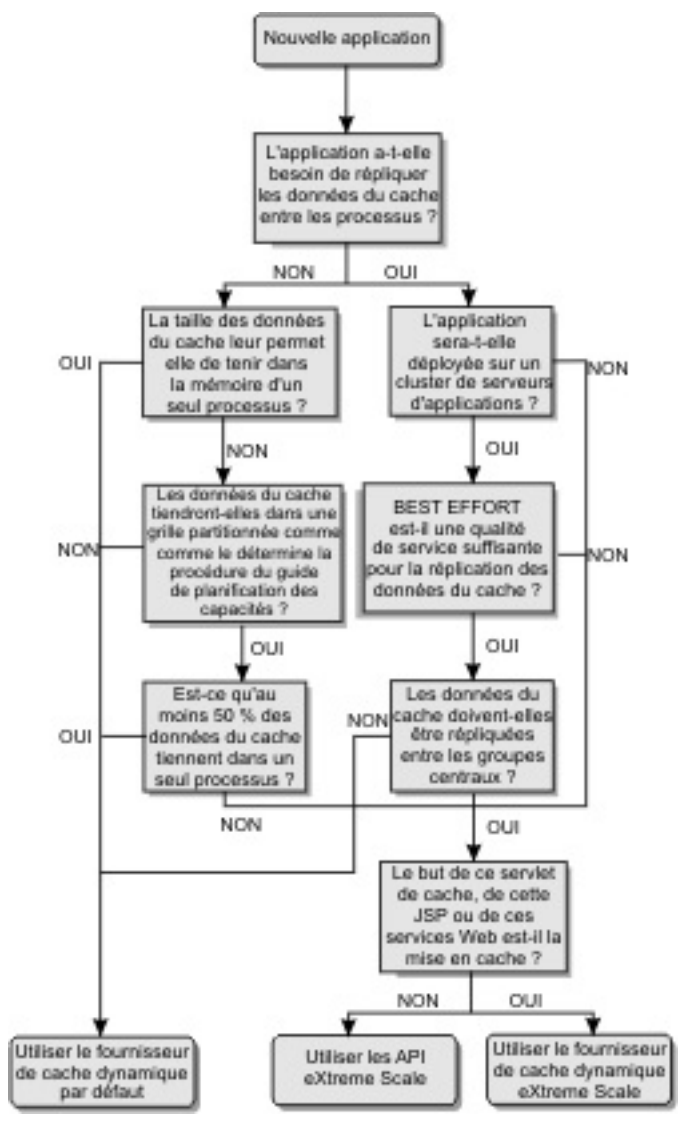
Les fonctions disponibles dans WebSphere DataPower XC10 Appliance améliorent sensiblement les fonctionnalités distribuées du service de cache dynamique par rapport au fournisseur de cache dynamique par défaut et les service de réplication de données. Avec eXtreme Scale, vous pouvez créer des mémoires cache véritablement réparties entre plusieurs serveurs et non simplement répliquées et synchronisées d'un serveur à l'autre. Par ailleurs, les mémoires cache eXtreme Scale sont transactionnelles et hautement disponibles : chaque serveur voit ainsi le même contenu pour le service de cache dynamique. WebSphere DataPower XC10 Appliance offre une qualité de service supérieure pour la réplication de cache fournie via DRS.

Tous ces avantages ne signifient cependant pas que le fournisseur de cache dynamique eXtreme Scale constitue la meilleure solution pour toutes les applications. Pour identifier la technologie la mieux adaptée à votre application, utilisez l'arbre de décision et la matrice de comparaison des fonctions.

Arbre de décision permettant de faire migrer des applications existantes de cache dynamique



Arbre de décision permettant de choisir un fournisseur de cache pour les nouvelles applications.



Comparaison des fonctionnalités

Tableau 1. Comparaison des fonctionnalités

Fonctionnalités de cache	Fournisseur par défaut	Fournisseur eXtreme Scale	API eXtreme Scale
Mise en cache local en mémoire	Oui	Via le cache local	Via le cache local
Mise en cache réparti	Via DRS	Oui	Oui

Evolutivité linéaire	Non	Oui	Oui
Réplication fiable (synchrone)	Non	Oui	Oui
Dépassement de capacité des disques	Oui	N/A	N/A
Expulsion	LRU/TTL/en fonction des segments	LRU/TTL (par partition)	LRU/TTL (par partition)
Invalidation	Oui	Oui	Oui
Relations	Relations d'ID de modèle/dépendance	Oui	Non (d'autres relations sont possibles)
Recherches non-clés	Non	Non	Via l'interrogation et l'index
Intégration dorsale	Non	Non	Via les chargeurs
Transactionnel	Non	Oui	Oui
Stockage à base de clés	Oui	Oui	Oui
Evénements et programmes d'écoute	Oui	Non	Oui
Intégration à WebSphere Application Server	Une seule cellule	Plusieurs cellules	Indépendant des cellules
Prise en charge de Java Standard Edition	Non	Oui	Oui
Surveillance et statistiques	Oui	Oui	Oui
Sécurité	Oui	Oui	Oui

Remarque : Le cache eXtreme Scale réparti peut uniquement stocker des entrées dont la clé et la valeur implémentent toutes les deux l'interface java.io.Serializable.

Moteur de cache dynamique et différences fonctionnelles avec eXtreme Scale

Les utilisateurs ne constatent aucune différence, sinon que les caches WebSphere DataPower XC10 Appliance ne supportent pas le déchargement sur disque ni les statistiques ou opérations en rapport avec la taille du cache en mémoire.

Il n'existe pas de différence notable dans les résultats retournés par la plupart des appels d'API de cache, que vous utilisiez le fournisseur de cache dynamique par défaut ou le fournisseur de cache eXtreme Scale.

Pour certaines opérations, il est impossible d'émuler le comportement du moteur de cache dynamique à l'aide d'eXtreme Scale.

Statistiques du cache dynamique

Appels des beans gérés

Le fournisseur de cache dynamique WebSphere eXtreme Scale ne prend pas en charge la mise en cache sur un disque. Les appels de beans gérés relatifs à une mise en cache sur un disque ne fonctionnent pas.

Mappage des règles de réplication du cache dynamique

La topologie distante du fournisseur de cache dynamique eXtreme Scale prend en charge une règle de réplication qui est très similaire aux règles SHARED_PULL et SHARED_PUSH_PULL (dans la terminologie utilisée par le fournisseur de cache dynamique WebSphere Application Server par défaut). Dans un cache dynamique eXtreme Scale, l'état distribué du cache est cohérent entre tous les serveurs.

Invalidation de l'index global

Vous pouvez utiliser un index global pour améliorer l'efficacité de l'invalidation dans les grands environnements partitionnés comportant, par exemple, plus de 40 partitions. Sans l'index global, le modèle de cache dynamique et le traitement de l'invalidation de dépendance doivent envoyer des demandes d'agent distant à toutes les partitions, ce qui affecte les performances. Lorsque vous configurez un index global, des agents d'invalidation sont envoyés uniquement aux partitions concernées qui contiennent des entrées de cache associées à l'ID de modèle ou de dépendance. Les possibilités d'amélioration des performances seront plus importantes dans les environnements comportant un grand nombre de partitions configurées. Vous pouvez configurer un index global en utilisant les index d'ID de dépendance et d'index qui sont disponibles dans les exemples de fichiers XML descripteurs objectGrid de cache dynamique.

Cache local

Vous pouvez configurer une instance de cache dynamique pour créer et gérer un cache local qui résidera dans la machine JVM du serveur d'applications. Ce cache contient un sous-ensemble des entrées contenues dans l'instance de cache dynamique distant. Pour plus d'informations, voir Configuration d'un cache local pour la mémoire cache dynamique. Il existe des propriétés personnalisées qui permettent d'optimiser le cache local. Pour plus d'informations, voir Propriétés personnalisées de cache dynamique.

2.5+

Réplication multimaître

Vous pouvez choisir d'activer la réplication multimaître dans le fournisseur de cache dynamique WebSphere DataPower XC10 Appliance. Pour plus d'informations, voir Création de grilles de données en cache dynamique.

Informations supplémentaires

- Redbook relatif au cache dynamique
- Documentation relative au cache dynamique
    - WebSphere Application Server 7.0
- Documentation relative à DRS
    - WebSphere Application Server 7.0

Rubrique parent : Java Création de grilles de données en cache dynamique

Java

Création de domaines de service de catalogue dans WebSphere Application Server

A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données de cache dynamique.

Avant de commencer

- La création de domaine de service de catalogue n'est requis que pour les grilles de données à caches dynamiques. Si vous utilisez des grilles de données simples ou des grilles de données de session, vous n'avez pas besoin de configurer de domaine de service de catalogue. WebSphere eXtreme Scale Client doit être installé sur WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

En créant un domaine de service de catalogue, vous définissez une collection de serveurs de catalogue à haute disponibilité. En configurant un domaine de service de catalogue, vous établissez des connexions aux serveurs de catalogue exécutés sur WebSphere DataPower XC10 Appliance. Ce domaine de service de catalogue représente le groupe des serveurs de catalogue qui s'exécutent sur les dispositifs de votre collectivité.

Procédure

1. Créez le domaine de service de catalogue.
    a. Dans la console d'administration de WebSphere Application Server, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaines de services de catalogue** > **Nouveau**.
    b. Définissez un nom, une valeur par défaut et des justificatifs d'identification pour l'authentification JMX de votre domaine.
    c. Ajoutez des points de contact de serveurs de catalogue.

       Spécifiez un groupe de serveurs distants qui sont les serveurs de catalogue qui s'exécutent sur les dispositifs de la collectivité. Pour visualiser les serveurs de catalogue qui s'exécutent dans la collectivité, cliquez sur **Collectivité** > **Membres** > ***nom_membre***. La zone **Serveurs de catalogue** affiche la liste des serveurs de catalogue en cours d'exécution dans la collectivité. Vous devez spécifier les points de contact avec leurs adresses IP ou leurs noms d'hôtes qualifiés complets. Utilisez les valeurs de port suivantes pour les dispositifs :
        - **Port de client** : non requis pour les connexions aux serveurs de catalogue exécutés sur le dispositif.
        - **Port d'écoute** : `2809`
2. Testez la connexion aux serveurs de catalogue dans le domaine de service de catalogue.
    a. Dans la console d'administration de WebSphere Application Server, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaines de services de catalogue**.
    b. Sélectionnez le domaine que vous voulez tester et cliquez sur **Tester la connexion**. Lorsque vous cliquez sur ce bouton, tous les points de contact des domaines de service de catalogue définis sont interrogés l'un après l'autre (s'il existe des points de contact) et la procédure retourne un message indiquant que la connexion au domaine a réussi.

Java **Tâches d'administration des domaines de service de catalogue**

Les langages de script Jacl ou Jython permettent de gérer les domaines de service de catalogue présents dans votre configuration WebSphere Application Server. A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données à caches dynamiques.

Rubrique parent : Java Création de grilles de données en cache dynamique

Java

Tâches d'administration des domaines de service de catalogue

Les langages de script Jacl ou Jython permettent de gérer les domaines de service de catalogue présents dans votre configuration WebSphere Application Server. A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données à caches dynamiques.

Conditions requises

WebSphere eXtreme Scale Client doit être installé dans votre environnement WebSphere Application Server.

Afficher la liste de toutes les tâches d'administration

Pour obtenir la liste de toutes les tâches d'administration associées aux domaines de service de catalogue, exécutez la commande suivante avec **wsadmin**:

```
wsadmin>$AdminTask help XSDomainManagement
```

Commandes

Les tâches d'administration de domaines de service de catalogue comprennent les commandes suivantes :

- createXSDomain
- deleteXSDomain
- getDefaultXSDomain
- listXSDomains
- modifyXSDomain
- getTransport
- testXSDomainConnection
- testXSServerConnection

createXSDomain

La commande **createXSDomain** enregistre un nouveau domaine de service de catalogue.

Tableau 1. Arguments de la commande createXSDomain

Argument	Description
-name (requis)	Spécifie le nom du domaine de service de catalogue à créer.
-default	Indique si le domaine de service de catalogue est le domaine par défaut de la cellule. La valeur par défaut est `true`. (booléen : a soit la valeur `true`, soit la valeur `false`).
-properties	Spécifie les propriétés personnalisées du domaine de service de catalogue.
-enableXIO	Indique si IBM eXtreme IO (XIO) ou ORB (Object Request Broker) est utilisé pour la communication de transport dans ce domaine de service de catalogue. **true** Indique que XIO est utilisé. **false** Indique qu'ORB est utilisé. Si vous n'indiquez pas de valeur, la valeur par défaut est `true` (XIO activé). Si le domaine de service de catalogue contient des serveurs distants, le paramètre **-enableXIO** ne configure pas XIO ni ORB sur les serveurs distants. Pour configurer le transport sur les serveurs distants, définissez le type de transport lorsque vous les démarrez.

Tableau 2. Arguments de la procédure defineDomainServers

Argument	Description
name_o	Spécifie le nom du point de contact du service de catalogue. • **Pour les serveurs d'applications existants** : le nom du noeud final doit avoir le format *cell name**node name**server name*

f_endpoint	• **Pour les serveurs distantes** : définit le nom d'hôte du serveur distant. Vous pouvez utiliser le même nom pour plusieurs noeuds finaux, mais les valeurs de port client doivent être uniques pour chaque noeud final.
custom_properties	Spécifie les propriétés personnalisées du point de contact du domaine de service de catalogue. Si vous ne disposez pas de propriétés personnalisées, utilisez des guillemets doubles ("") pour cet argument.
endpoint_ports	Spécifie les numéros de port du point de contact du domaine de service de catalogue. Les ports doivent être définis dans l'ordre suivant : *<client_port>,<listener_port>* **Port du client** Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent uniquement dans des processus WebSphere Application Server et elle peut correspondre à n'importe quel port inutilisé autre part. **Port d'écoute** Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue. **Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance** : utilisez la valeur 2809 pour les noeuds finaux distants d'appliance.

Valeur retournée :

Exemples de mode de traitement par lots

Le mode de traitement par lots impose de formater correctement l'entrée de commande. Utilisez le mode interactif pour que les valeurs que vous entrez soient correctement traitées. Lorsque vous utilisez le mode de traitement par lots, vous devez définir les arguments d'étape **-defineDomainServers** en utilisant un tableau de propriétés spécifiques. Ce tableau a le format *name_of_endpoint custom_properties endpoint_ports*. La valeur *endpoint_ports* est la liste des ports qui doivent être définis dans l'ordre suivant : *<client_port>,<listener_port>*.

- Créez un domaine de service de catalogue de noeuds finaux distants en utilisant Jacl :

```
$AdminTask createXSDomain {-name TestDomain -default true -defineDomainServers
{{xhost1.ibm.com "" ,2809}} }}
```

- Créez un domaine de service de catalogue de noeuds finaux distants en utilisant la chaîne Jython :

```
AdminTask.createXSDomain('[-name TestDomain -default true
-defineDomainServers [[xhost1.ibm.com "" ,2809]
[xhost2.ibm.com "" ,2809]]  ]')
```

- réez un domaine de service de catalogue de noeuds finaux de serveur d'applications existants en utilisant Jacl :

```
$AdminTask createXSDomain {-name TestDomain -default true -defineDomainServers
 {{cellName/nodeName/serverName "" 1109}}}
```

Exemples de mode interactif

- Jacl :

```
$AdminTask createXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.createXSDomain ('[-interactive]')
```

deleteXSDomain

La commande **deleteXSDomain** supprime un domaine de service de catalogue.

Paramètres requis :

-name

Spécifie le nom du domaine de service de catalogue à supprimer.

Valeur retournée :

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask deleteXSDomain {-name TestDomain }
```

- A l'aide de la chaîne Jython :

```
AdminTask.deleteXSDomain('[-name TestDomain ]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask deleteXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.deleteXSDomain ('[-interactive]')
```

getDefaultXSDomain

La commande **getDefaultXSDomain** retourne le domaine de service de catalogue par défaut de la cellule.

Paramètres requis : aucun.

Valeur retournée : le nom du domaine de service de catalogue par défaut.

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask getDefaultXSDomain
```

- A l'aide de la chaîne Jython :

```
AdminTask.getDefaultXSDomain
```

Exemples de mode interactif

- Jacl :

```
$AdminTask getDefaultXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.getDefaultXSDomain ('[-interactive]')
```

listXSDomains

La commande **listXSDomains** retourne la liste des domaines de service de catalogue existants.

Paramètres requis : aucun.

Valeur retournée : la liste de tous les domaines de service de catalogue présents dans la cellule.

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask listXSDomains
```

- A l'aide de la chaîne Jython :

```
AdminTask.listXSDomains
```

Exemples de mode interactif

- Jacl :

```
$AdminTask listXSDomains {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.listXSDomains ('[-interactive]')
```

modifyXSDomain

La commande **modifyXSDomain** modifie un domaine de service de catalogue existant.

Le mode de traitement par lots impose de formater correctement l'entrée de commande. Utilisez le mode interactif pour que les valeurs que vous entrez soient correctement traitées. Lorsque vous utilisez le mode de traitement par lots, vous devez définir les arguments d'étape **-modifyEndpoints**, **-addEndpoints** et **-removeEndpoints** en utilisant un tableau de propriétés spécifiques. Ce tableau a le format *name_of_endpoint host_name custom_properties endpoint_ports*. La valeur *endpoint_ports* est la liste des ports qui doivent être définis dans l'ordre suivant : *<client_port>*,*<listener_port>*.

Tableau 3. Arguments de la commande modifyXSDomain

Argument	Description
-name (requis)	Spécifie le nom du domaine de service de catalogue que vous souhaitez éditer.
-default	Avec la valeur `true`, spécifie que le domaine de service de catalogue est le domaine par défaut de la cellule (booléen).
-properties	Spécifie les propriétés personnalisées du domaine de service de catalogue.
-enableXIO	Indique si IBM eXtreme IO (XIO) ou ORB (Object Request Broker) est utilisé pour la communication de transport dans ce domaine de service de catalogue. **true** Indique que XIO est utilisé. **false** Indique qu'ORB est utilisé. Si vous n'indiquez pas de valeur, la valeur par défaut est `true` (XIO activé). Si le domaine de service de catalogue contient des serveurs distants, vous ne pouvez pas configurer XIO sur les serveurs distants.

Tableau 4. Arguments de la procédure modifyEndpoints

Argument	Description
name_of_endpoint	Spécifie le nom du point de contact du service de catalogue. • **Pour les serveurs d'applications existants** : le nom du noeud final doit avoir le format *cell_name**node_name**server_name* • **Pour les serveurs distantes** : définit le nom d'hôte du serveur distant. Vous pouvez utiliser le même nom pour plusieurs noeuds finaux, mais les valeurs de port client doivent être uniques pour chaque noeud final. Cette valeur doit être un nom qualifié complet de domaine si vous configurez un dispositif.
endpoint_ports	Spécifie les numéros de port du point de contact du domaine de service de catalogue. Les noeuds finaux doivent être définis dans l'ordre suivant : *<client_port>*, *<listener_port>* **Port du client**

ts	Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent uniquement dans des processus WebSphere Application Server et elle peut correspondre à n'importe quel port inutilisé autre part. **Port d'écoute** Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue. **Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance** : utilisez la valeur 2809 pour les noeuds finaux distants d'appliance.

Tableau 5. Arguments de la procédure addEndpoints

Argument	Description
name_of_endpoint	Spécifie le nom du point de contact du service de catalogue. • **Pour les serveurs d'applications existants** : le nom du noeud final doit avoir le format *cell_name**node_name**server_name* • **Pour les serveurs distantes** : définit le nom d'hôte du serveur distant. Vous pouvez utiliser le même nom pour plusieurs noeuds finaux, mais les valeurs de port client doivent être uniques pour chaque noeud final. Cette valeur doit être un nom qualifié complet de domaine si vous configurez un dispositif.
custom_properties	Spécifie les propriétés personnalisées du point de contact du domaine de service de catalogue. Si vous ne disposez pas de propriétés personnalisées, utilisez des guillemets doubles ("") pour cet argument.
endpoint_ports	Spécifie les numéros de port du point de contact du domaine de service de catalogue. Les noeuds finaux doivent être définis dans l'ordre suivant : <*client_port*>, <*listener_port*> **Port du client** Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent uniquement dans des processus WebSphere Application Server et elle peut correspondre à n'importe quel port inutilisé autre part. **Port d'écoute** Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue. **Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance** : utilisez la valeur 2809 pour les noeuds finaux distants d'appliance.

Tableau 6. Arguments de la procédure removeEndpoints

Argument	Description
name_of_endpoint	Spécifie le nom du point de contact de domaine de service de catalogue à supprimer.

Valeur retournée :

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask modifyXSDomain {-name TestDomain -default true -modifyEndpoints
{{xhost1.ibm.com "" ,2809}} -addEndpoints {{xhost2.ibm.com "" ,2809}}}
-removeEndpoints {{xhost3.ibm.com}}}
```

- A l'aide de la chaîne Jython :

```
AdminTask.modifyXSDomain('[-name TestDomain
-default false -modifyEndpoints [[xhost1.ibm.com "" ,2809]]
-addEndpoints [[xhost3.ibm.com "" ,2809]]
-removeEndpoints [[xhost2.ibm.com]]]')
```

- Modifiez un domaine de service de catalogue existant pour activer IBM eXtremeIO :

```
AdminTask.modifyXSDomain('[-name testDomain -enableXIO true]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask modifyXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.modifyXSDomain ('[-interactive]')
```

getTransport

La commande **getTransport** affiche le type de transport pour le domaine de service de catalogue : IBM eXtremeIO (XIO) ou ORB (Object Request Broker). Si vous exécutez cette commande sur un domaine de service de catalogue qui contient des serveurs distants ou que catalogServerName est un serveur distant, une erreur se produit. Vous devez utiliser la commande **xscmd -c showTransport** pour les serveurs distants.

Paramètres requis :

-domainName

Spécifie le nom du domaine de service de catalogue pour lequel vous voulez afficher le type de transport.

-catalogServerName

Spécifie le nom du serveur de catalogue pour lequel vous voulez afficher le type de transport.

Valeur de retour : ORB ou XIO

Affichage du type de transport d'un domaine de service de catalogue

- Jacl :

```
$AdminTask getTransport {-domainName TestDomain }
```

- A l'aide de la chaîne Jython :

```
AdminTask.getTransport('[-domainName testDomain]')
```

Affichage du transport d'un serveur de catalogue

-
- Jacl :

```
$AdminTask getTransport {-catalogServerName myCell01\myNode01\container1 }
```

- A l'aide de la chaîne Jython :

```
AdminTask.getTransport('[-catalogServerName myCell01\myNode01\container1]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask getTransport {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.getTransport ('[-interactive]')
```

testXSDomainConnection

La commande **testXSDomainConnection** teste la connexion à un domaine de service de catalogue.

Paramètres requis :

-name

Spécifie le nom du domaine de service de catalogue vers lequel tester la connexion.

Paramètres facultatifs

-timeout

Spécifie en secondes pendant combien de temps au maximum attendre la connexion.

Valeur retournée : `true` s'il est possible d'établir une connexion, sinon, une erreur de connexion est retournée.

Exemples de mode de traitement par lots

- Jacl :

```
$Admintask testXSDomainConnection
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSDomainConnection
```

Exemples de mode interactif

- Jacl :

```
$AdminTask testXSDomainConnection {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSDomainConnection ('[-interactive]')
```

testXSServerConnection

La commande **testXSServerConnection** teste la connexion à un serveur de catalogue. Cette commande fonctionne aussi bien pour les serveurs autonomes que pour les serveurs qui font partie d'un domaine de service de catalogue.

Paramètres requis :

host

Spécifie l'hôte sur lequel réside le serveur de catalogue.

listenerPort

Spécifie le port d'écoute du serveur de catalogue.

Paramètres facultatifs

timeout

Spécifie en secondes pendant combien de temps au maximum attendre la connexion au serveur de catalogue.

Valeur retournée :

Exemples de mode de traitement par lots

- Jacl :

```
$Admintask testXSServerConnection {-host xhost1.ibm.com -listenerPort 2809}
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask testXSServerConnection {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSServerConnection ('[-interactive]')
```

Rubrique parent : Java Création de domaines de service de catalogue dans WebSphere Application Server

Java

Configuration d'un cache local pour la mémoire cache dynamique

2.5+ Vous pouvez configurer un cache local afin d'utiliser une grille de données de mémoire cache dynamique sur le dispositif ou la collectivité. Le cache local utilise des ressources JVM locales. Généralement, le cache local comporte un sous-ensemble des données qui figurent dans la grille de données de cache dynamique sur le dispositif.

Avant de commencer

Créez une grille de données de mémoire cache dynamique. Pour plus d'informations, voir Création de grilles de données en cache dynamique.

Pourquoi et quand exécuter cette tâche

Pour activer le cache local d'une grille de données de mémoire cache dynamique, vous devez définir une propriété personnalisée pour définir le nom du modèle de mappe pour l'instance de cache dynamique WebSphere Application Server.

Procédure

Définissez la propriété personnalisée suivante pour activer le cache local :

- com.ibm.websphere.xs.dynacache.map_template_name : Indiquez `IBM_DC_NCI_PARTITIONED_.*` pour modifier le nom de modèle. La définition de cette propriété personnalisée pour une instance de cache active un cache local pour cette instance de cache.

Pour plus d'informations sur la définition de propriétés personnalisées pour la mémoire cache dynamique, voir Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées.

Que faire ensuite

Par défaut, la taille maximum du cache local dynamique correspond au nombre maximum d'entrées pour une seule partition dans la grille de données du cache dynamique distant. Un algorithme d'expulsion LRU (least recently used) est utilisé pour gérer la taille. Pour mieux contrôler la taille du cache local, vous pouvez effectuer les actions suivantes :

- Configurez la propriété personnalisée com.ibm.websphere dynacache.near_cache_size pour spécifier le nombre maximum d'entrées de cache admises dans le cache local. Pour plus d'informations, voir Propriétés personnalisées de cache dynamique.

Rubrique parent : Java Création de grilles de données en cache dynamique

Configuration de la capacité maximale d'une grille de données

Vous pouvez définir une capacité maximale pour chaque grille de données de la collectivité. La configuration d'une capacité maximale limite la quantité de stockage de données qui peut être utilisée par une grille de données. La limite de capacité permet de s'assurer que la capacité de stockage disponible pour la collectivité est utilisée de façon prévisible.

Avant de commencer

- Créez les grilles de données pour votre configuration. Par défaut, les grilles de données ne sont pas configurées avec une limite de capacité maximale. Vous pouvez configurer une capacité maximale pour tous les types de grilles de données : grilles de données simples, grilles de données de session ou grilles de données de mémoire cache dynamique.

Pourquoi et quand exécuter cette tâche

Après la configuration de limites de capacité maximale sur chaque grille de données de la collectivité, la limite de capacité est appliquée en comparant la taille totale de toutes les données principales de la grille de données à la limite de capacité configurée pour la grille de données. La capacité utilisée par les copies de réplique des données n'est pas comptée lorsque la grille de données est mesurée par rapport à la limite de capacité configurée.

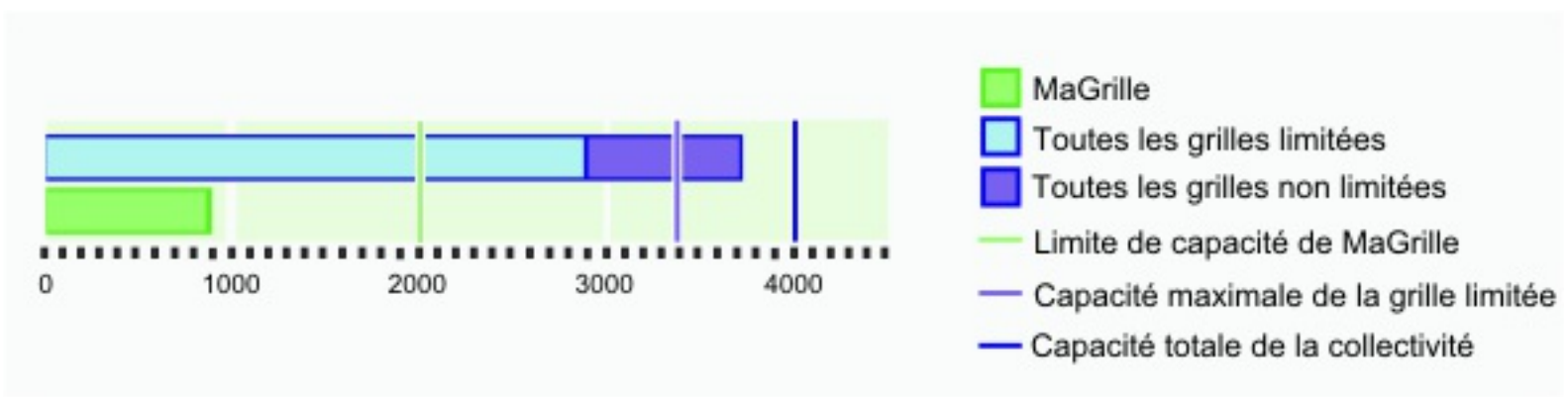
La limite de capacité maximale est une quantité maximale de données qui peut être insérée dans la grille de données. La limite ne constitue pas la garantie d'une quantité d'espace alloué pour la grille de données. Par conséquent, une grille de données peut ne pas atteindre sa limite de capacité configurée si la collectivité n'a pas la capacité pour stocker les données. Les raisons d'une capacité insuffisante dans la collectivité peuvent être une limite de capacité élevée sur la grille de données ou une capacité qui est consommée par d'autres grilles de données de la collectivité.

Lorsque la limite de capacité pour une grille de données particulière est dépassée, la grille traite les opérations d'insertion de l'une des manières suivantes :

- Par défaut, les opérations d'insertion faisant dépasser la capacité limite de la grille sont rejetées. Les processus client reçoivent une exception en réponse aux demandes d'insertion. Les opérations de lecture, de mise à jour et de suppression réussissent même si la grille de données dépasse sa limite de capacité. Avec ces opérations limitées, la grille de données peut s'exécuter à un niveau fonctionnel minimal, mais un accroissement supplémentaire de la grille de données est interdit.
- Disponible uniquement sur une grille de données simple, vous pouvez remplacer ce comportement par défaut en sélectionnant l'option **Expulsion la moins récemment utilisée pour cette grille de données**. Le fait de sélectionner cette option permet de nouvelles insertions dans une grille de données simple et maintient la limite de capacité en supprimant l'entrée de données la moins récemment utilisée. La définition de la capacité maximale sur une grille de données ne nécessite pas de redémarrage ; cependant, si vous avez sélectionné l'option d'expulsion la moins récemment utilisée sur une grille simple, cette dernière est automatiquement redémarrée pour que les modifications prennent effet. Il en est de même si vous décidez de désactiver l'option d'expulsion la moins récemment utilisée sur une grille simple.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > ***type_grille_de_données*** > ***nom_grille_de_données*** > **Afficher les attributs avancés**.
2. Sélectionnez **Limiter la quantité de capacité pour cette grille de données**.
3. Si vous définissez la capacité maximale pour une grille de données simplet et que vous voulez que la grille accepte de nouvelles opérations d'insertion (au lieu de les rejeter) aux dépens des entrées de données les moins récemment utilisées, sélectionnez l'option **Expulsion la moins récemment utilisée**. Cliquez sur **Appliquer les modifications** pour enregistrer les modifications. Vous êtes prévenu que les données de la grille seront perdues pour que le redémarrage aboutisse.
4. Affichez la consommation de capacité actuelle pour déterminer la capacité maximale à définir pour la grille de données sélectionnée. Vous pouvez aussi vous assurer que vous ne dépassez pas la capacité totale de la collectivité.



Dans ce graphique, la grille de données à configurer, MaGrille, utilise actuellement 900 Mo de capacité. Elle a une limite de capacité actuellement configurée à 2000 Mo. Au niveau de la collectivité, la capacité totale est de 4000 Mo. En outre, le total des toutes les limites configurées sur les grilles de données limitées en capacité est de 3400 Mo. Ces grilles utilisent actuellement 2900 Mo. Enfin, au moins une des grille de données de la collectivité n'a pas de limite de capacité définie. Ces grilles de données sans limite de capacité définie consomment environ 900 Mo.

5. Entrez une valeur pour la limite de consommation des données principales en Mo. Lorsque vous appuyez sur Entrée, la consommation de capacité maximale potentielle des données principales et des données répliquées s'affiche. Ce nombre varie en fonction du nombre de répliques que vous avez définies. Rappelez-vous cependant que le nombre de répliques est limité par le nombre de dispositifs de la collectivité. Si vous avez quatre répliques définies et qu'il y a trois dispositifs dans la collectivité, votre collectivité comprend une entité de données principales et deux répliques.
6. Cliquez sur **Appliquer les modifications** pour sauvegarder la configuration. Il n'est pas nécessaire de redémarrer votre grille de données pour activer la nouvelle limite.

Exemple

Exemple de limite de capacité : Plusieurs grilles de données

Des grilles de données A, B et C sont définies dans une collectivité avec une capacité de stockage totale de 600 Go. Aucune réplique n'est définie sur les grilles de données. La grille de données A a une limite de capacité de 100 Go. La grille de données B a une limite de capacité de 50 Go. La grille de données C a une limite de capacité de 200 Go. Dans ce scénario, au moins 250 Go de capacité non utilisée sont toujours disponibles dans la collectivité. La taille totale des trois grilles de données ne peut pas croître au-delà de 350 Go.

Exemple de limite de capacité : Répliques

La grille de données A est définie dans une collectivité de deux dispositifs. La grille de données A a une réplique synchrone et deux répliques asynchrones, pour un total de trois répliques. La limite de capacité de la grille est définie à 100 Mo. A l'origine, la consommation de capacité maximale de cette grille est de 200 Mo. La collectivité ayant seulement deux dispositifs, il n'existe qu'une copie principale et une copie de réplique. La grille de données principale peut utiliser jusqu'à 100 Mo. La réplique grandit dans les mêmes proportions que la grille de données principale, ce qui aboutit à une capacité consommée totale maximale de 200 Mo. Si un troisième dispositif est ajouté à la collectivité, une seconde copie de réplique est mise en place. La consommation maximale de la grille passe à 300 Mo, pour la grille principale plus les deux répliques.

Exemple de limite de capacité : Grilles de données sans limite de capacité

Des grilles de données A, B et C sont définies dans une collectivité avec une capacité de stockage totale de 600 Go. La grille de données A a une limite de capacité de 100 Go. La grille de données B a une limite de capacité de 50 Go. La grille de données C n'a pas de limite de capacité. Aucune réplique n'est définie pour aucune des trois grilles. La grille de données C n'ayant pas de limite, la grille de données peut potentiellement consommer la totalité des 600 Go de capacité disponible. Par conséquent, la grille de données A et la grille de données B ne pourraient plus insérer de données. Les données insérées par la grille de données A ou la grille de données B sont conservées, mais il n'est pas garanti que les grilles de données puissent atteindre leur limite de capacité. La grille de données C est sûre d'avoir au moins 450 Go disponibles à consommer car les seules autres grilles de données du système ne peuvent pas consommer plus d'un total de 150 Go sur les 600 Go de capacité. Ce calcul de 450 Go ignore les capacités qui sont consommées par les données répliquées. Si deux grilles de données non limitées ou plus existent dans la collectivité, la capacité potentielle d'une grille de données spécifique n'est pas garantie.

Rubrique parent : Configuration des grilles de données

Effacement des grilles de données

Vous pouvez supprimer définitivement toutes les entrées d'une grille de données. Vous pouvez effacer la grille de données pour supprimer les informations périmées ou tester les entrées.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > ***data_grid_type*** > ***data_grid_name***.
2. Cliquez sur l'icône d'effacement de grille () pour supprimer toutes les entrées de grille de données. Vous devez confirmer la suppression de toutes les entrées dans la grille de données.
3. Vous pouvez vérifier que les entrées de grille de données ont été supprimées de l'interface utilisateur. Cliquez sur **Surveiller** > **Vue d'ensemble des domaines de grilles de données** > ***data_grid_name*** et affichez le graphique **Capacités utilisées vs. Nombre d'entrées en cache**. Pour les grilles de données simples et les grilles de données de session, le nombre d'entrées dans la grille de données doit être proche de zéro. Toutefois, avec une grille de données de cache dynamique, plusieurs entrées restent dans la grille de données. Ces entrées de grille de données contiennent les informations de configuration de la grille de données de cache dynamique.

Rubrique parent : Configuration des grilles de données

Suppression des grilles de données

Si vous souhaitez supprimer les données d'une grille de données, vous pouvez supprimer la grille de données puis la recréer.

Pourquoi et quand exécuter cette tâche

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > ***type_grille_données***. Sélectionnez le nom ***nom_grille_données*** à supprimer.
2. Cliquez sur l'icône de suppression ( ) pour lancer la procédure de suppression. Un message s'affiche alors pour vous inviter à confirmer la suppression définitive de cette grille de données. Cliquez sur **OK** pour confirmer la suppression.
3. Vous pouvez contrôler la suppression de la grille de données dans la vue **Tâches**.

Résultats

Rubrique parent : Configuration des grilles de données

Java

Configuration d'un fournisseur de cache Spring

Spring Framework Version 3.1 a introduit une nouvelle abstraction de cache. Celle-ci vous permet d'ajouter de manière transparente la mise en cache à une application Spring existante. Vous pouvez utiliser WebSphere DataPower XC10 Appliance comme fournisseur de cache pour l'abstraction de cache.

Avant de commencer

- Vous devez disposer d'une application utilisant Spring Framework version 3.1 ou ultérieure.
- Votre application doit déclarer les méthodes de mise en cache à l'aide d'annotations. Pour plus d'informations sur la mise à jour de votre application pour l'abstraction de cache, voir Spring Framework Reference Documentation : Cache abstraction.
- Assurez-vous que le fichier `ogclient.jar` se trouve dans le chemin d'accès aux classes de l'application Spring.
- Si la machine virtuelle Java (JVM) sur laquelle s'exécute votre application n'est pas celle installée par WebSphere eXtreme Scale Client, vous devez ajouter l'argument JVM suivant de sorte que l'IBM Object Request Broker (ORB) soit utilisé :

  ```
  -Djava.endorsed.dirs=wxs_root/lib/endorsed
  ```

- Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.
- Lors de la connexion de l'application Spring à des grilles de données sécurisées, vous devez spécifier un fichier `client.properties` approprié comme valeur du paramètre **`client-security-config`**. Spécifiez ce paramètre dans l'ObjectGridCatalogServiceDomainBean de la configuration de conteneur Spring Inversion of Control (IoC). Vous pouvez configurer le fournisseur de cache Spring de sorte qu'il utilise l'authentification client ainsi que TLS pour le transport de réseau sécurisé. Pour plus d'informations, voir Fichier de propriétés du client, Configuration d'une application de grille de données pour l'utilisation de l'authentification client et Configuration de TLS pour les applications de grille de données.
- Vous devez connaître le nom d'hôte du serveur de catalogue et le port du dispositif. Pour obtenir ces informations, cliquez sur **Collectivité** > **Membres** dans l'interface utilisateur.

Pourquoi et quand exécuter cette tâche

A l'aide de l'abstraction de cache dans Spring Framework, vous pouvez réduire le nombre d'exécutions de la méthode d'application Spring. Lorsqu'elle est configurée, les résultats d'une méthode particulière sont placés dans la mémoire cache. Quand la méthode est ré-exécutée, l'abstraction vérifie la mémoire cache pour déterminer si les résultats de la méthode y figurent déjà. Si les résultats sont dans le cache, les résultats sont renvoyés à partir de la mémoire cache et la méthode ne s'exécute pas. La mise en oeuvre de l'abstraction peut donc réduire le nombre d'exécutions des méthodes coûteuses, ce qui réduit également le temps de réponse moyen de l'application.

Procédure

Configurez le conteneur Spring IoC (Inversion of Control) de sorte qu'il utilise WebSphere DataPower XC10 Appliance comme fournisseur de cache. L'implémentation de cache WebSphere DataPower XC10 Appliance réside dans le package com.ibm.websphere.objectgrid.spring. Définissez les beans ci-dessous dans votre configuration de conteneur Spring IoC.

```
<bean id="wxsCSDomain"
class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean"
  p:catalog-service-endpoints="CATALOG_SERVICE_ENDPOINTS"
  />

<bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
  p:object-grid-name="OBJECT_GRID_NAME"
  p:catalog-service-domain-ref="wxsCSDomain" />

<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
  <property name="caches">
    <set>
      <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
         p:name="CACHE_NAME"
         p:map-name="MAP_NAME "
                p:object-grid-client-ref="wxsGridClient" />
     </set>
   </property>
```

```
</bean>
```

CATALOG_SERVICE_ENDPOINTS

Indique le nom d'hôte et le port du serveur de catalogue.
Indique le chemin d'accès absolu ou relatif à un fichier XML ObjectGrid dans lequel modifier les paramètres côté client sous la forme d'une ressource Spring. Pour plus d'informations sur la spécification des ressources dans Spring, voir Spring Framework Reference Documentation: Resources.

Exemple : `p:client-override-xml="file:/path/to/objectgrid.xml"`

Exemple : `p:client-override-xml="classpath:com/example/app/override-objectgrid.xml"`

Exemple : `p:client-override-xml="http://myserver/override-objectgrid.xml"`

Exemple : `p:client-override-xml="ftp://myserver/override-objectgrid.xml"`

CLIENT_SECURITY_CONFIG **(facultatif)**

Indique le chemin d'accès absolu ou relatif à un fichier `client.properties` sous la forme d'une ressource Spring. Pour plus d'informations sur la spécification des ressources dans Spring, voir Spring Framework Reference Documentation: Resources. Pour plus d'informations sur la création d'un fichier `client.properties` pour WebSphere DataPower XC10 Appliance, voir Configuration d'une application de grille de données pour l'utilisation de l'authentification client.

Exemple : `p:client-security-config="file:/path/to/client.properties"`

OBJECT_GRID_NAME

Spécifie le nom d'ObjectGrid. Ce paramètre n'est pas requis si les serveurs de conteneur sont démarrés à l'aide des fichiers de configuration XML fournis. Cette valeur correspond au nom de la grille de données simple que vous avez créée dans l'interface utilisateur.

CACHE_NAME

Indique le nom de la mémoire cache spécifiée dans l'application de mise en cache Spring.

MAP_NAME

Indique le nom de la mappe de sauvegarde pour une mémoire cache. Cette valeur correspond au nom de la grille de données simple que vous avez créée dans l'interface utilisateur. Pour utiliser un nom de mappe autre que la valeur par défaut, vous pouvez définir une mappe dynamique. Pour plus d'informations sur la création de mappes dynamiques, voir Options de configuration de mappe dynamique.

Exemple

Le fragment de code ci-dessous crée une mémoire cache intitulée `default`, hébergée par un dispositif dans `myXC10.myhost.com:2809`. Cet exemple utilise l'instance de mappe par défaut qui est nommée d'après la grille de données.

```
<bean id="wxsCSDomain"
class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean"
        p:catalog-service-endpoints ="myXC10.myhost.com:2809" />
<bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
        p:object-grid-name=”my_simple_data_grid”
  p:catalog-service-domain-ref="wxsCSDomain" />
<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
        <property name="caches">
                <set>
                        <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
                                p:name="default"
                                p:map-name=”my_simple_data_grid”
                                p:object-grid-client-ref="wxsGridClient" />
                </set>
        </property>
</bean>
```

Rubrique parent : Configuration des grilles de données

Configuration des clients

Vous pouvez configurer certaines propriétés sur les clients en remplaçant les propriétés définies sur les serveurs. Vous pouvez remplacer ces propriétés dans le fichier des propriétés du client ou à l'aide d'un programme.

Configuration des clients Java

Vous pouvez configurer WebSphere eXtreme Scale pour l'exécuter dans un environnement autonome ou dans un environnement avec WebSphere Application Server. Pour qu'un déploiement WebSphere eXtreme Scale sélectionne les modifications de configuration dans la grille de serveurs, vous devez redémarrer les processus pour que ces modifications entrent en vigueur au lieu d'être appliquées de manière dynamique. Toutefois, côté client, vous ne pouvez pas modifier les paramètres de configuration d'une instance de client existante, mais vous pouvez créer une instance de client avec les paramètres nécessaires en utilisant un fichier XML ou à l'aide d'un programme. Lorsque vous créez un client, vous pouvez remplacer les paramètres par défaut provenant de la configuration de serveur actuelle.

.NET 2.5+ **Configuration de WebSphere eXtreme Scale Client for .NET**

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET à l'aide du fichier de propriétés du client, de la configuration XML côté serveur, ou en remplaçant à l'aide d'un programme certaines propriétés du serveur.

Java

Configuration des clients Java

Vous pouvez configurer WebSphere eXtreme Scale pour l'exécuter dans un environnement autonome ou dans un environnement avec WebSphere Application Server. Pour qu'un déploiement WebSphere eXtreme Scale sélectionne les modifications de configuration dans la grille de serveurs, vous devez redémarrer les processus pour que ces modifications entrent en vigueur au lieu d'être appliquées de manière dynamique. Toutefois, côté client, vous ne pouvez pas modifier les paramètres de configuration d'une instance de client existante, mais vous pouvez créer une instance de client avec les paramètres nécessaires en utilisant un fichier XML ou à l'aide d'un programme. Lorsque vous créez un client, vous pouvez remplacer les paramètres par défaut provenant de la configuration de serveur actuelle.

Vous pouvez configurer un client eXtreme Scale (client Java uniquement) au moyen des méthodes suivantes, chacune pouvant être effectuée avec un fichier XMLde remplacement sur le client ou à l'aide d'un programme :

- Configuration XML
- Configuration par programmation
- Configuration Spring Framework
- Désactivation du cache local

Remplacements sur le client Java
Vous pouvez configurer un client WebSphere eXtreme Scale en fonction de vos besoins en remplaçant les paramètres serveur. Vous pouvez remplacer plusieurs plug-ins et attributs.

Configuration des clients Java avec une configuration XML
Vous pouvez utiliser le fichier XML de configuration pour modifier les paramètres du client.

Configuration des clients Java à l'aide d'un programme
Vous pouvez remplacer les paramètres à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

Définition du délai d'attente pour les requêtes et les nouvelles tentatives
Vous pouvez définir des options d'optimisation afin de contrôler pendant combien de temps le code du client eXtreme Scale attend la fin de l'exécution d'une requête d'accès à la grille de données ou renouvelle ses tentatives d'exécution d'une requête de ce type.

Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session
Si des applications client utilisent la gestion de session et sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour qu'il utilise la grille de données du dispositif pour la gestion de session.

Déploiement d'une passerelle REST
Vous pouvez déployer et configurer la passerelle REST pour la grille de données dans WebSphere Application Server ou dans un serveur profil Liberty.

Rubrique parent : Configuration des clients

Java

Remplacements sur le client Java

Vous pouvez configurer un client WebSphere eXtreme Scale en fonction de vos besoins en remplaçant les paramètres serveur. Vous pouvez remplacer plusieurs plug-ins et attributs.

Pour remplacer les paramètres sur un client, vous pouvez utiliser XML ou la configuration par programmation. Pour plus d'informations sur le remplacement des paramètres client, voir Configuration des clients Java avec une configuration XML et Configuration des clients Java à l'aide d'un programme.

Vous pouvez remplacer les plug-in suivants sur un client :

- **BackingMap**
  - Plug-in Evictor
  - Plug-in MapEventListener
  - Plug-in BackingMapLifecycleListener
  - Plug-in MapSerializerPlugin
- **Attributs BackingMap**
  - Attribut numberOfBuckets

    Obsolète : La propriété est obsolète. Utilisez l'attribut nearCacheEnabled pour activer le cache local.

  - attribut timeToLive
  - attribut ttlEvictorType
  - attribut evictionTriggers
  - attribut nearCacheEnabled
  - attribut nearCacheInvalidationEnabled
  - attribut nearCacheLastAccessTTLSyncEnabled
- **ObjectGrid**
  - Plug-in TransactionCallback
  - Plug-in ObjectGridEventListener
  - Plug-in ObjectGridLifecycleListener
- **attribut ObjectGrid**
  - attribut entityMetadataXMLFile
  - attribut txTimeout
  - attribut txIsolation

Rubrique parent : Configuration des clients Java

Java

Configuration des clients Java avec une configuration XML

Vous pouvez utiliser le fichier XML de configuration pour modifier les paramètres du client.

Pourquoi et quand exécuter cette tâche

Pour modifier les paramètres d'un client WebSphere eXtreme Scale, créez un fichier XML ObjectGrid dont la structure est similaire à celle du fichier utilisé pour le serveur de conteneur.

Pour la liste des modules d'extension et des attributs que vous pouvez remplacer sur le client, voir Remplacements sur le client Java.

Procédure

1. Créez un fichier XML de configuration ObjectGrid pour le client qui ait une structure semblable à celle du fichier pour le serveur de conteneur.

Supposons que le fichier XML a été associé à un fichier XML de stratégie de déploiement et que ces fichier ont été utilisés pour démarrer un serveur de conteneur.

companyGridServerSide.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyTxCallback" />
            <bean id="ObjectGridEventListener"
                className="com.company.MyOgEventListener" />
            <backingMap name="Customer"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" nearCacheEnabled="true"
                timeToLive="1600" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor"
/>
            <bean id="MapEventListener"
                className="com.company.MyMapEventListener" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Sur un serveur de conteneur, l'instance ObjectGrid nommée CompanyGrid se comporte conformément à ce qui est défini dans le fichier `companyGridServerSide.xml`. Par défaut, les paramètres du client CompanyGrid sont identiques à ceux de l'instance CompanyGrid qui s'exécute sur le serveur.

Le fichier XML ObjectGrid suivant peut être utilisé pour définir certains attributs et plug-in du client CompanyGrid.

```
companyGridClientSide.xml

<?xml version="1.0" encoding="UTF-8"?>
```

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyClientTxCallback" />
            <bean id="ObjectGridEventListener" className="" />
            <backingMap name="Customer" nearCacheEnabled="true"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" nearCacheEnabled="true"
                timeToLive="800" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor"
/>
            <bean id="MapEventListener" className="" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Le fichier XML spécifie les remplacements suivants :

- Le bean TransactionCallback sur le client est com.company.MyClientTxCallback au lieu du paramètre côté serveur com.company.MyTxCallback.
- Le client n'est associé à aucun plug-in ObjectGridEventListener car la valeur className est la chaîne vide.
- Le client active un cache local pour Customer backingMap, conserve son plug-in Evictor et supprime le plug-in MapEventListener.
- L'attribut timeToLive d'OrderLine backingMap a changé.
- Bien qu'un attribut lockStrategy différent ait été indiqué, les conséquences sont nulles car cet attribut n'est pas pris en charge pour un remplacement par le client.

2. Créez le client avec le fichier XML.

   Pour créer le client CompanyGrid à l'aide du fichier `companyGridClientSide.xml`, transmettez le fichier XML ObjectGrid sous la forme d'une URL à l'une des méthodes de connexion dans l'interface ObjectGridManager :

   ```
   ObjectGridManager ogManager =
           ObjectGridManagerFactory.ObjectGridManager();
   ClientClusterContext clientClusterContext =
           ogManager.connect("MyServer1.company.com:2809", null, new URL(
                   "file:xml/companyGridClientSide.xml"));
   ```

Rubrique parent : Configuration des clients Java

Java

Configuration des clients Java à l'aide d'un programme

Vous pouvez remplacer les paramètres à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

Pourquoi et quand exécuter cette tâche

L'exemple de code suivant crée les mêmes substitutions que celles décrites dans Configuration des clients Java avec une configuration XML.

Pour la liste des modules d'extension et des attributs que vous pouvez remplacer sur le client, voir Remplacements sur le client Java.

Procédure

Le code suivant crée une instance ObjectGrid côté client.

```
ObjectGridConfiguration companyGridConfig = ObjectGridConfigFactory
    .createObjectGridConfiguration("CompanyGrid");
Plugin txCallbackPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.TRANSACTION_CALLBACK, "com.company.MyClientTxCallback");
companyGridConfig.addPlugin(txCallbackPlugin);

Plugin ogEventListenerPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.OBJECTGRID_EVENT_LISTENER, "");
companyGridConfig.addPlugin(ogEventListenerPlugin);

BackingMapConfiguration customerMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("Customer");
customerMapConfig.setNumberOfBuckets(1429);
Plugin evictorPlugin = ObjectGridConfigFactory.createPlugin(PluginType.EVICTOR,
    "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor");
customerMapConfig.addPlugin(evictorPlugin);

companyGridConfig.addBackingMapConfiguration(customerMapConfig);

BackingMapConfiguration orderLineMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("OrderLine");
orderLineMapConfig.setNumberOfBuckets(701);
orderLineMapConfig.setTimeToLive(800);
orderLineMapConfig.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);

companyGridConfig.addBackingMapConfiguration(orderLineMapConfig);

List ogConfigs = new ArrayList();
ogConfigs.add(companyGridConfig);

Map overrideMap = new HashMap();
overrideMap.put(CatalogServerProperties.DEFAULT_DOMAIN, ogConfigs);

ogManager.setOverrideObjectGridConfigurations(overrideMap);
ClientClusterContext client = ogManager.connect(catalogServerEndpoints, null, null);
ObjectGrid companyGrid = ogManager.getObjectGrid(client, objectGridName);
```

L'instance ogManager de l'interface ObjectGridManager recherche uniquement les remplacements dans les objets ObjectGridConfiguration et BackingMapConfiguration que vous incluez dans la mappe overrideMap. Par exemple, le code précédent remplace le nombre de compartiments de la mappe OrderLine. La mappe Order reste cependant inchangée côté client car aucune configuration de cette mappe n'est incluse.

Rubrique parent : Configuration des clients Java

Java

Définition du délai d'attente pour les requêtes et les nouvelles tentatives

Vous pouvez définir des options d'optimisation afin de contrôler pendant combien de temps le code du client eXtreme Scale attend la fin de l'exécution d'une requête d'accès à la grille de données ou renouvelle ses tentatives d'exécution d'une requête de ce type.

Pourquoi et quand exécuter cette tâche

Vous pouvez configurer des paramètres du client eXtreme Scale qui contrôlent pendant combien de temps le client tente de créer des connexions réseau, de traiter une requête de grille de données pour une partition et de renouveler cette requête avant de renvoyer une exception à votre application.

Facteurs d'optimisation du délai d'attente pour les requêtes et les nouvelles tentatives pour XIO et ORB

Pour certaines options d'optimisation, l'emplacement de définition des délais d'attente dépend du transport utilisé : eXtremeIO (XIO) ou Object Request Broker (ORB). Ces options de niveau transport sont les premières à avoir une incidence sur les interactions avec votre client, car elles déterminent pendant combien de temps le transport tente d'établir des connexions socket réseau et de combien de temps un appel de procédure éloignée (RPC, remote procedure call) analogue à une opération de grille de données dispose pour son exécution.

Lorsque vous optimisez ces valeurs, tenez compte de ce que votre environnement peut tolérer dans des conditions de charge maximum et d'état stabilisé. Si vous définissez les intervalles trop en dessous des valeurs par défaut (30 secondes pour le délai d'attente pour les nouvelles tentatives, par exemple), vos opérations risquent d'échouer prématurément. Tenez compte des facteurs suivants :

- Temps d'attente réseau
- Couplage des interactions de la grille avec des ressources externes telles que des bases de données
- Pauses de la récupération de place qui résultent de votre combinaison de politiques d'optimisation de taille de segment de mémoire, d'utilisation de segment de mémoire et de récupération de place

Paramètres ORB d'optimisation des délais d'attente pour les requêtes et les nouvelles tentatives

Pour ORB, les paramètres de délai d'attente sont les suivants :

com.ibm.CORBA.ConnectionTimeout

Délai pendant lequel ORB tente de créer une connexion socket avec l'emplacement distant. ORB place ces connexions en cache, ce qui fait que cette opération n'est pas répétée pour chaque requête.

com.ibm.CORBA.RequestTimeout

Délai pendant lequel ORB attend la fin de l'exécution d'un appel de procédure éloignée.

com.ibm.CORBA.FragmentTimeout

Pour plus de détails, voir la documentation IBM consacrée à ORB. Le produit est fourni avec une valeur par défaut pour cette valeur.

com.ibm.CORBA.LocateRequestTimeout

Pour plus de détails, voir la documentation IBM consacrée à ORB. Le produit est fourni avec une valeur par défaut pour cette valeur.

Lorsque vous optimisez les paramètres RequestTimeout et ConnectionTimeout, il peut être intéressant de les ajuster en tenant compte des recommandations par défaut. Vous pouvez également leur affecter la même valeur, basée sur le délai d'attente souhaité pour les requêtes.

Le niveau d'optimisation suivant concerne le paramètre requestRetryTimeout. Pour chaque type de transport, après qu'il a émis une exception système car un appel de procédure éloignée n'a pas abouti dans le délai imparti, la grille de données peut utiliser le délai supplémentaire défini par le paramètre requestRetryTimeout (par exemple, le délai d'attente pour les requêtes peut être de 10 secondes et le délai d'attente pour les nouvelles tentatives, de 20 secondes) pour définir pendant combien de temps elle effectue les actions suivantes :

- Envoi au serveur de catalogue d'une demande asynchrone en vue d'obtenir la table de routage la plus récente, pour le cas où les partitions seraient situées ailleurs en raison d'un basculement.
- Utilisation de nouvelles routes et nouvelle tentative d'exécution de la requête, ou arrêt des tentatives et envoi d'une exception à votre application.

La propriété requestRetryTimeout est définie en millisecondes. Une valeur supérieure à zéro indique que la demande doit être renouvelée lors de l'émission d'une exception pour laquelle une nouvelle tentative est possible. Une valeur de `0` indique qu'aucune nouvelle tentative ne doit avoir lieu en cas d'exception. Pour utiliser le comportement par défaut, supprimez la propriété ou attribuez-lui la valeur `-1`.

Paramètres XIO d'optimisation des délais d'attente pour les nouvelles tentatives

Pour XIO, les paramètres consolidés suivants sont disponibles :

- Le paramètre xioTimeout détermine le délai pendant lequel le transport XIO tente d'établir une connexion socket réseau.
- Les paramètres ORB LocateRequest et FragmentTimeout n'ont pas d'équivalent.
- Le paramètre requestRetryTimeout contrôle le délai maximum dont un appel de procédure éloignée (RPC) dispose pour aboutir, et il aide à contrôler pendant combien de temps de nouvelles tentatives d'appel RPC ont lieu lorsque de nouvelles informations de routage sont obtenues du serveur de catalogue. Lorsque vous utilisez le transport XIO, il signale souvent beaucoup plus tôt que ORB qu'un appel de procédure éloignée est sur le point d'échouer. Cet échec se produit avant l'expiration du délai d'attente requestRetryTimeout.

Comment définir le délai d'attente pour les nouvelles tentatives

Le délai d'attente pour les nouvelles tentatives d'exécution des requêtes peut être défini dans le fichier des propriétés du client ou dans une session. La valeur définie dans la session remplace la valeur qui figure dans les propriétés du client. Si la valeur définie est supérieure à zéro, la demande est renouvelée jusqu'à ce que le délai d'attente expire ou qu'une erreur permanente se produise. Une erreur permanente peut être une exception `DuplicateKeyException`. La valeur zéro définit le mode "fail-fast" et la grille de données ne retente pas la transaction, quel que soit son type.

Délai d'attente pour les transactions et délai d'attente pour les nouvelles tentatives d'exécution de requête

Pendant l'exécution, le délai d'attente pour les transactions est utilisé avec le délai d'attente pour les nouvelles tentatives d'exécution de requête, ce qui garantit que ce dernier ne dépasse pas le délai d'attente pour les transactions.

Deux types de transaction existent : les transactions à validation automatique et les transactions qui utilisent des méthodes explicites begin et commit. Les exceptions qui autorisent de nouvelles tentatives diffèrent pour ces deux types de transaction :

- Les transactions appelées dans une session sont retentées pour ORB CORBA SystemException (TransportException pour XIO) et les exceptions TargetNotAvailable du client eXtreme Scale.
- Les transactions à validation automatique sont retentées pour CORBA SystemException et les exceptions de disponibilité du client eXtreme Scale. Ces dernières sont ReplicationVotedToRollbackTransactionException, TargetNotAvailable et AvailabilityException.

Les erreurs d'application et autres erreurs permanentes provoquent l'émission immédiate d'une exception et le client ne retente pas la transaction. Ces erreurs permanentes incluent les exceptions DuplicateKeyException et KeyNotFoundException. Utilisez le paramètre "fail-fast" pour émettre toutes les exceptions sans retenter les transactions.

Exceptions pour lesquelles le client retente la transaction :

- ReplicationVotedToRollbackTransactionException (uniquement en validation automatique)
- TargetNotAvailable
- org.omg.CORBA.SystemException (TransportException est l'équivalent XIO de cette exception système ORB)
- AvailabilityException (uniquement en validation automatique)
- LockTimeoutException (uniquement en validation automatique)
- UnavailableServiceException (uniquement en validation automatique)

Exceptions permanentes pour lesquelles la transaction n'est pas retentée :

- DuplicateKeyException
- KeyNotFoundException
- LoaderException
- TransactionAffinityException
- LockDeadlockException
- OptimisticCollisionException

Procédure

- Définition du délai d'attente pour les nouvelles tentatives dans un fichier de propriétés de client.

  Pour définir la valeur de requestRetryTimeout dans un client, ajoutez ou modifiez cette propriété dans le Fichier de propriétés du client. Par défaut, le fichier dans lequel les propriétés du client sont définies est le fichier `objectGridClient.properties`. La propriété requestRetryTimeout est définie en millisecondes. Une valeur supérieure à zéro indique que la demande doit être renouvelée lors de l'émission d'une exception pour laquelle une nouvelle tentative est possible. Une valeur de `0` indique qu'aucune nouvelle tentative ne doit avoir lieu en cas d'exception. Pour utiliser le comportement par

défaut, supprimez la propriété ou attribuez-lui la valeur `-1`. Voici un exemple de valeur dans le fichier `objectGridClient.properties` :

```
requestRetryTimeout = 30000
```

La valeur de requestRetryTimeout est spécifiée en millisecondes. Dans l'exemple, si la valeur est utilisée dans une instance ObjectGrid, la valeur de requestRetryTimeout sera de 30 secondes.

- Définition du délai d'attente pour les nouvelles tentatives à l'aide d'un programme.

  Pour définir les propriétés du client à l'aide d'un programme, commencez par créer un fichier de propriétés dans un <`emplacement`> approprié pour votre application. Dans l'exemple ci-dessous, le fichier des propriétés du client est celui cité dans le fragment de code de la section précédente (`objectGridClient.properties`). Après vous être connecté à une instance ObjectGridManager, définissez les propriétés du client comme indiqué. Ensuite, lorsque vous avez une instance ObjectGrid, cette instance possède les propriétés client que vous avez définies dans le fichier. Chaque fois que vous serez amené à modifier ce fichier, vous devrez explicitement obtenir une nouvelle instance ObjectGrid.

  ```
  ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
  String objectGridName = “testObjectGrid”;
  URL clientXML = null;
  ClientClusterContext ccc = manager.connect("localhost:2809", null, clientXML);
  File file = new File("<location>/objectGridClient.properties");
  URL url = file.toURI().toURL();
  ccc.setClientProperties(objectGridName, url);
  ObjectGrid objectGrid = ogManager.getObjectGrid(ccc, objectGridName);
  ```

- Définissez le fichier de substitution pendant une validation de session.

  Pour définir dans l'objet Session pendant combien de temps il convient d'effectuer de nouvelles tentatives, ou pour remplacer la propriété client requestRetryTimeout, appelez la méthode setRequestRetryTimeout(long) dans l'interface Session.

  ```
  Session sessionA = objectGrid.getSession();
  sessionA.setRequestRetryTimeout(30000);
  ObjectMap mapA = sessionA.getMap("payroll");
  String key = "key:" + j;
  mapA.insert(key, "valueA");
  ```

  Cette session utilise à présent la valeur 30 000 millisecondes (30 secondes) pour requestRetryTimeout, quelle que soit la valeur définie dans le fichier des propriétés du client. Pour plus d'informations sur l'interface de session, voir Utilisation des sessions pour accéder aux données de la grille.

Exemple

Dans l'exemple ci-dessous, le client peut traiter le temps d'attente réseau, la récupération de place et les conflits généraux sur le serveur grâce à la définition de délais d'attente courts. La propriété **`requestRetryTimeout`** est définie sur 10 secondes, et la propriété **`xioTimeout`** est définie comme la valeur ORB **`ConnectionTimeout`**, c'est-à-dire sur 5 secondes.

Tableau 1. Configurations de grille de données pour les types de transport ORB et eXtremeIO

Type de grille	**ORB**	**XIO**
Une application client Java™ ou .NET qui accède directement à une API eXtreme Scale	• Modifiez le fichier `orb.properties` de votre application client. Définissez les valeurs suivantes : ◦ com.ibm.CORBA.RequestTimeout=5 ◦ com.ibm.CORBA.ConnectTimeout=5 ◦ com.ibm.CORBA.FragmentTimeout=5 ◦ com.ibm.CORBA.LocateRequestTimeout=5 **Remarque :** Avec WebSphere Application Server, vous contrôlez les paramètres ORB à l'aide du gestionnaire de déploiement et non pas du fichier `orb.properties`. • Modifiez le fichier `objectGridClient.properties` de	Modifiez le fichier `objectGridClient.properties` de votre application client en indiquant les valeurs suivantes : • xioRequestTimeout=50000. Cette valeur représente des millisecondes et correspond au paramètre com.ibm.CORBA.RequestTime

	`objectGridClient.properties` de l'application client en indiquant requestRetryTimeout=7000.	out. • xioTimeout=5. Cette valeur représente des secondes et correspond au paramètre com.ibm.CORBA.ConnectTimeout. • requestRetryTimeout=70000. Cette valeur représente des millisecondes et est également utilisée pour le transport ORB. • ORB FragmentTimeout et LocateRequestTimeout n'ont pas d'équivalent en XIO.
Session HTTP	Placez le fichier de propriétés du client dans le chemin d'accès aux classes, en y définissant la valeur `requestRetryTimeout=10000`.	Modifiez le fichier de propriétés de client de votre application client en indiquant les valeurs suivantes : • xioRequestTimeout=10000. Cette valeur représente des millisecondes et correspond au paramètre com.ibm.CORBA.RequestTimeout. • xioTimeout=5. Cette valeur représente des secondes et correspond au paramètre com.ibm.CORBA.ConnectTimeout. • requestRetryTimeout=10000. Cette valeur représente des millisecondes et est également utilisée pour le transport ORB. • ORB FragmentTimeout et LocateRequestTimeout n'ont pas

		d'équivalent en XIO.
Cache dynamique	Définissez la propriété suivante dans l'instance de mémoire cache : • com.ibm.websphere.xs.dynacache.request_retry_timeout_override=2000 Si vous souhaitez que la valeur de cette propriété d'instance de cache soit utilisée pour toutes les instances de mémoire cache dynamique, vous pouvez, au lieu d'utiliser cette propriété, utiliser le paramètre **requestRetryTimeout** en plaçant le fichier de propriétés du client dans le chemin d'accès aux classes.	

Rubrique parent : Configuration des clients Java

Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Si des applications client utilisent la gestion de session et sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour qu'il utilise la grille de données du dispositif pour la gestion de session.

Avant de commencer

Cette tâche inclut les instructions qui permettent de déployer une application qui requiert la gestion de session. Pour savoir comment créer une application Java qui utilise la gestion de session, voir Développement d'applications de grilles de données avec des API Java.

Pourquoi et quand exécuter cette tâche

De même que vous pouvez configurer vos applications WebSphere Application Server pour qu'elles utilisent le dispositif pour la gestion de session, vous pouvez configurer le profil Liberty à cette fin.

Vous pouvez décider d'utiliser le profil Liberty avec le dispositif si vous avez besoin d'un serveur léger offrant des fonctions dynamiques. Par exemple, vous pouvez ajouter ou supprimer des fonctions, lesquelles sont des entités qui permettent de contrôler les éléments de l'environnement d'exécution chargés sur un serveur donné. Par conséquent, dans le profil Liberty, si vous exécutez des applications qui gèrent les sessions, par exemple, vous pouvez créer une définition de serveur que vous utilisez pour spécifier des fonctions du profil Liberty, lesquelles contrôlent la façon dont le serveur interagit avec la grille de données du dispositif.

Procédure

1. Créez une définition de serveur en exécutant la commande suivante :

   ```
   répertoire_install_wlp/bin/server create nom_serveur
   ```

2. Recherchez le fichier `server.xml` sous la définition du serveur et ouvrez-le dans un éditeur XML.
3. Placez l'application de session (par exemple, *votre_application*`.jar`) dans le répertoire `/wlp/usr/servers/defaultServer/apps`.
4. Démarrez la console de surveillance du dispositif, puis cliquez sur **Grille de données** > **Grille de session**.
5. Créez sur le dispositif une grille de session nommée `session`.
6. Exportez la propriété **`JAVA_HOME`** à partir d'une ligne de commande. Veillez à exécuter la commande à partir du répertoire dans lequel est installé le profil Liberty.
7. Démarrez le profil Liberty à l'aide de la commande suivante :

   ```
   ./server start servername
   ```

   Un PID s'affiche.
8. Ouvrez l'application de session à l'aide de l'URL suivante :

   ```
   http://server:port/A/
   ```

9. Exécutez des tests de session pour vérifier que les données sont bien écrites dans la grille de session du dispositif.

Fichier de définition de serveur sans SSL activé

Voici l'exemple d'un fichier `server.xml` de base pour lequel SSL n'est pas activé. L'exemple suivant est présenté sur plusieurs lignes en raison des contraintes liées à la publication.

Remarque : La fonction Web est obsolète. Utilisez la fonction webApp à la place. Lorsque vous ajoutez la fonction Web à la définition de serveur et configurez le gestionnaire de sessions, vous pouvez utiliser la réplication de session dans les applications WebSphere® eXtreme Scale exécutées dans le profil Liberty.

Etudiez l'exemple suivant, dans lequel la fonction webApp est utilisée :

2.5+

```
<server description="new server">

    <!-- Enable features -->
```

```
    <featureManager>
        <feature>jsp-2.2</feature>
      <feature>eXtremeScale.server-1.1</feature>
      <feature>eXtremeScale.webApp-1.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />

    <xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809"
securityEnabled="false" />
    <applicationMonitor updateTrigger="mbean"/>
    <application id="A" location="A.ear" name="A" type="ear"/>
     <httpSession cloneId="A_test"/>

   <!-- <com.ibm.ws.xs.server.config catalogServer="true" /> -->
</server>
```

Configuration du profil Liberty pour les grilles de données qui font l'objet d'un accès avec SSL activé
Si des applications client sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour HTTPS, lequel utilise automatiquement SSL et le chiffrement de données pour les serveurs Web sécurisés.

Configuration du profil Liberty pour une exécution avec des clients
Utilisez la fonction du client WebSphere eXtreme Scale pour exécuter le profil Liberty avec des clients eXtreme Scale.

Activation de la fonction webApp dans le profil Liberty
Un serveur de profil Liberty peut héberger une grille de données qui place en mémoire cache les données des applications en vue de la réplication des données de session HTTP pour la tolérance aux pannes.

Propriétés de la fonction xsDynacacheApp du profil Liberty
Spécifiez le profil Liberty qui doit héberger la grille de données, que vous pouvez configurer comme fournisseur de cache dynamique à l'aide de cette fonction.

Rubrique parent : Configuration des clients Java

Configuration du profil Liberty pour les grilles de données qui font l'objet d'un accès avec SSL activé

Si des applications client sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour HTTPS, lequel utilise automatiquement SSL et le chiffrement de données pour les serveurs Web sécurisés.

Procédure

1. Exécutez la commande suivante pour créer le certificat SSL et activer le protocole HTTPS :

   ```
   cd to lib_dir\bin
   securityUtility createSSLCertificate --server=defaultServer --password=xc10test
   ```

2. Ajoutez la fonction SSL suivante au fichier `server.xml` afin de configurer le profil Liberty pour une exécution du chiffrement de données SSL :
   2.5+

   ```
   <featureManager>
           <feature>ssl-1.1</feature>
       </featureManager>
       <keyStore id="defaultKeyStore" password="{xor}MjowbTI+Kyw=" />
   ```

3. Démarrez le profil Liberty à l'aide de la commande suivante :

   ```
   ./server start servername
   ```

   Un PID s'affiche.

4. Ouvrez l'application de session à l'aide de l'URL suivante :

   ```
   http://server:securedport/A/
   ```

5. Exécutez des tests de session pour vérifier que les données sont écrites dans la grille de session du dispositif.

Exemple de fichier de définition activé par SSL

Certaines lignes de code sont réparties sur plusieurs lignes pour des raisons de mise en page. Voici l'exemple d'une configuration de fichier `server.xml` avancée qui utilise la fonction SSL. L'exemple suivant est présenté sur plusieurs lignes en raison des contraintes de mise en page.

Remarque : A compter de la version 2.5, le numéro de version de fonction `webApp-1.0` devient `webApp-1.1`.

2.5+

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>eXtremeScale.server-1.1</feature>
      <feature>eXtremeScale.webApp-1.1</feature>
       <feature>ssl-1.1</feature>
    </featureManager>

      <httpEndpoint id="defaultHttpEndpoint"
                    host="*"
                    httpPort="9080"
                    httpsPort="9443">
        <!--tcpOptions soReuseAddr="true" / -->
    </httpEndpoint>

    <keyStore id="defaultKeyStore" password="{xor}Jzxubys6LCs=" />

 <xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809"
securityEnabled="true"
credentialGeneratorClass="com.ibm.websphere.objectgrid.security.plugins.builtins.UserPassw
ordCredentialGenerator"
```

```
   credentialGeneratorProps="xcadmin xcadmin"/>
    <applicationMonitor updateTrigger="mbean"/>
    <application id="A" location="A.ear" name="A" type="ear"/>
    <httpSession cloneId="A_test"/>

</server>
```

Que faire ensuite

Pour définir la configuration SSL entre le profil Liberty et le conteneur de la grille de données, spécifiez le type de transport du client dans le fichier de propriétés du client. Les valeurs possibles sont :

- **`TCP/IP`** : Indique que le client ne prend en charge que les connexions TCP/IP.
- **`SSL pris en charge`** : Indique que le client prend en charge les connexions TCP/IP et SSL (Secure Sockets Layer). (Valeur par défaut)
- **`SSL requis`**: Indique que le client exige des connexions SSL.

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Tâches associées:
Configuration de la sécurité du client

Configuration du profil Liberty pour une exécution avec des clients

Utilisez la fonction du client WebSphere eXtreme Scale pour exécuter le profil Liberty avec des clients eXtreme Scale.

Avant de commencer

Effectuez les tâches suivantes avant de configurer le profil Liberty :

- Installez le profil Liberty et WebSphere eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Cette configuration ne fournit que la fonction client. La fonction serveur s'exécute dans un autre processus. L'ajout de la fonction client permet à l'application d'accéder aux API eXtreme Scale et de se connecter à une grille distante.

Cette configuration client fournit un processus unique incluant ce dont vous avez besoin pour exécuter une application Web à l'aide d'une grille de données eXtreme Scale. Après avoir ajouté la fonction client, l'application peut écrire dans les API eXtreme Scale.

Procédure

N'ajoutez la fonction client qu'au serveur du profil Liberty. Ajoutez le code suivant au serveur du profil Liberty :

```
eXtremeScale.client-1.1
```

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Activation de la fonction webApp dans le profil Liberty

Un serveur de profil Liberty peut héberger une grille de données qui place en mémoire cache les données des applications en vue de la réplication des données de session HTTP pour la tolérance aux pannes.

Pourquoi et quand exécuter cette tâche

Lorsque vous installez le WebSphere Application Server profil Liberty, il ne contient pas la réplication de session. Cependant, si vous utilisez la grille de données avec le profil Liberty, vous pouvez répliquer les sessions pour que les utilisateurs de l'application ne perdent pas les données de session en cas de défaillance d'un serveur.

Lorsque vous ajoutez la fonction Web à la définition de serveur et configurez le gestionnaire de sessions, vous pouvez utiliser la réplication de session dans les applications de grille de données exécutées dans le profil Liberty.

Procédure

Ajoutez la fonction webApp ci-dessous au fichier profil Liberty `server.xml`. La fonction webApp inclut la fonction client, mais pas la fonction serveur. Vous voudrez certainement séparer les applications Web des grilles de données. Par exemple, vous disposez d'un serveur profil Liberty pour vos applications Web et d'un autre serveur profil Liberty pour l'hébergement de la grille de données.

```
<featureManager>
<feature>eXtremeScale_webapp-1.1</feature>
</featureManager>
```

Résultats

Vos applications Web peuvent maintenant conserver leurs données de session dans une grille de données.

Exemple

Etudiez l'exemple de fichier `server.xml` suivant, qui contient la fonction Web que vous utilisez lorsque vous vous connectez à la grille de données à distance :

```
<server description="Airport Entry eXtremeScale Getting Started Client Web Server">
<!--
Ce programme exemple n'est soumis à aucune redevance ; il est fourni EN L'ETAT et peut
être librement utilisé, exécuté, copié et modifié
sans paiement de redevance par le client
(a) pour sa propre formation,
(b) pour développer des applications qui doivent s'exécuter avec un produit IBM WebSphere,
à des fins d'utilisation interne par le client pour que le client le redistribue dans le
cadre d'une telle application
dans les propres produits du client.
Licensed Materials - Property of IBM
5724-X67, 5655-V66 (C) COPYRIGHT International Business Machines Corp. 2012
-->
<!-- Enable features -->
<featureManager>
<feature>eXtremeScale.webapp-1.1</feature>
</featureManager>

<httpEndpoint id="defaultHttpEndpoint"
host="*"
httpPort="${default.http.port}"
httpsPort="${default.https.port}" />

<xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809"
securityEnabled="false" />

</server>
```

Que faire ensuite

La fonction webApp dispose de propriétés de métadonnées que vous pouvez définir dans l'élément xsWebApp du fichier `server.xml`. Pour plus d'informations, voir Propriétés de la fonction xsWebApp du profil Liberty.

[Propriétés de la fonction xsWebApp du profil Liberty]
Définissez la fonction webApp pour étendre l'application Web dans le profil Liberty. Ajoutez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Propriétés de la fonction xsWebApp du profil Liberty

Définissez la fonction webApp pour étendre l'application Web dans le profil Liberty. Ajoutez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

La fonction Web est obsolète. Utilisez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

Vous pouvez définir les attributs suivants dans l'élément xsWebApp du fichier `server.xml` :

Paramètres

`objectGridName`

Valeur de chaîne qui définit le nom de l'instance ObjectGrid utilisée pour une application Web particulière. Le nom par défaut est session.

Cette propriété doit refléter le nom objectGridName dans les fichiers XML ObjectGrid et XML de déploiement utilisés pour démarrer les serveurs de conteneur eXtreme Scale.

`catalogHostPort`

Le serveur de catalogues peut être contacté pour obtenir une instance ObjectGrid côté client. La valeur doit avoir le format `host:port<,hôte:port>`. L'hôte est le programme d'écoute sur lequel le serveur de catalogue s'exécute. Le port est le port d'écoute du processus serveur de catalogue. La longueur de cette liste peut être arbitraire et la liste n'est utilisée que pour l'amorçage. La première adresse viable qui est utilisée. Elle est facultative dans WebSphere Application Server si la propriété **`catalog.services.cluster`** est définie.

`replicationInterval`

Entier (en secondes) qui définit le temps séparant deux écritures de sessions actualisées vers la grille. La valeur par défaut est 10 secondes. Les valeurs possibles sont comprises entre 0 et 60. 0 signifie que les sessions actualisées sont écrites dans la grille pour chaque demande dès la fin de l'appel à la méthode de service du servlet. Une valeur `replicationInterval` plus élevée améliore les performances, car un moins grand nombre de mises à jour sont écrites dans la grille de données. Mais en même temps, une valeur supérieure à 0 rend la configuration moins tolérante aux pannes.

Ce paramètre s'applique uniquement lorsque objectGridType a la valeur `REMOTE`.

`sessionTableSize`

Entier qui définit le nombre de références de session conservées en mémoire. La valeur par défaut est `1000`.

Ce paramètre appartient uniquement à une topologie `REMOTE`, car la topologie `EMBEDDED` a déjà les données de session dans le même groupe que le conteneur Web.

Les sessions sont expulsées de la table interne en fonction de la logique LRU (least recently used). Lorsqu'une session est expulsée de cette table, elle est invalidée dans le conteneur Web. Cependant, les données ne sont pas pour autant supprimées de la grille, ce qui permet aux demandes ultérieures de cette session de continuer à extraire les données. Cette valeur doit être supérieure à la valeur maximale du pool d'unités d'exécution du conteneur Web, ce qui réduit les conflits au niveau du cache de session.

`fragmentedSession`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `true`. Ce paramètre permet de contrôler si le produit stocke les données de session en tant qu'entrée entière ou s'il stocke chaque attribut séparément.

Affectez au paramètre fragmentedSession la valeur `true` si la session d'application Web a de nombreux attributs ou des attributs avec des grandes tailles. Affectez à fragmentedSession la valeur `false` si une session a peu d'attributs, car tous les attributs sont stockés dans la même clé dans la grille de données.

Dans la précédente implémentation à base de filtres, il était fait référence à cette propriété en tant que mécanisme de persistance avec, comme valeurs possibles, ObjectGridStore (fragmentation) et ObjectGridAtomicSessionStore (non-fragmentation).

`securityEnabled`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Ce paramètre active la sécurité du client eXtreme Scale. Il doit correspondre au paramètre **`securityEnabled`** dans le fichier des propriétés sur serveur eXtreme Scale. Si les paramètres ne correspondent pas, une exception est générée.

credentialAuthentication

Indique si l'authentification des données d'identification est imposée ou prise en charge.

Jamais

Aucune authentification de certificat client n'est imposée.

Requise

L'authentification des données d'identification est toujours appliquée. Si le serveur ne prend pas en charge l'authentification des données d'identification, le client ne peut pas se connecter au serveur.

Pris en charge

(Par défaut) L'authentification des données d'identification est imposée seulement si à la fois le client et le serveur prennent en charge l'authentification des données d'identification.

authenticationRetryCount

Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe utilisée pour obtenir les données d'identification des clients. La valeur par défaut est `0`.

credentialGeneratorClass

Le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe sert à obtenir les données d'identification des clients.

credentialGeneratorProps

Les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés correspondent à l'objet avec la méthode setProperties(String). La valeur credentialGeneratorProps n'est utilisée que si la valeur de la propriété **credentialGeneratorClass** n'est pas null.

shareSessionsAcrossWebApps

Spécifie si les sessions sont partagées entre des applications Web ; spécifiée comme valeur de chaîne `true` ou `false`. La valeur par défaut est `false`. La spécification de servlet indique que les sessions HTTP ne peuvent pas être partagées entre des applications Web. Une extension à la spécification de servlet est fournie pour permettre ce partage.

Rubrique parent : Activation de la fonction webApp dans le profil Liberty

Propriétés de la fonction xsDynacacheApp du profil Liberty

Spécifiez le profil Liberty qui doit héberger la grille de données, que vous pouvez configurer comme fournisseur de cache dynamique à l'aide de cette fonction.

Vous pouvez définir les attributs suivants dans l'élément `xsDynacacheApp` du fichier `server.xml` :

Paramètres

`gridName`

Valeur de chaîne qui définit le nom de l'instance de grille de données utilisée pour une instance de cache dynamique particulière.

`cacheName`

Définit le nom du suffixe unique qui est utilisé comme nom du modèle de mappe. Par exemple :

```
IBM_DC_PARTITIONED.nom_cache
```

`mapName`

Définit le nom de la mappe de la grille de données qui joue le rôle de fournisseur de cache dynamique.

`remoteDomain`

Définit le nom de domaine client éloigné du fournisseur de cache dynamique de la grille de données dans le profil Liberty.

`clientObjectgridXML`

Définit l'emplacement du fichier `objectgrid.xml` client pour eXtreme Scale.

`requestRetryTimeout`

Définit le temps (en millisecondes) pendant lequel une requête peut s'exécuter avant qu'un dépassement de délai ne se produise. Cette propriété remplace la propriété **`requestRetryTimeout`** si elle est définie dans le fichier de propriétés du client. La valeur par défaut pour les instances de mémoire cache dynamique est de `2000` millisecondes.

`templateName`

Indique le nom du préfixe du modèle de mappe. Vous pouvez utiliser l'un des modèles de mappe suivants :

- `IBM_DC_PARTITIONED_.*` (valeur par défaut)
- `IBM_DC_NCI_PARTITIONED_.*` (indique que ce cache utilise un cache local)

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Déploiement d'une passerelle REST

Vous pouvez déployer et configurer la passerelle REST pour la grille de données dans WebSphere Application Server ou dans un serveur profil Liberty.

Avant de commencer

Vérifiez qu'un serveur profil Liberty est créé. Pour plus d'informations, voir Installation de profil Liberty.

Pourquoi et quand exécuter cette tâche

La passerelle REST est un servlet qui est défini dans le fichier archive Web (WAR) `wxsRESTGateway.war`. Avec cette passerelle REST, vous utilisez un identificateur URI (Uniform Resource Identifier) pour accéder aux données dans la grille de données.

Procédure

1. Activez la fonction de passerelle REST en modifiant manuellement le fichier `server.xml` ou en utilisant Liberty Profile Developer Tools.
   - Activez la passerelle REST dans le fichier profil Liberty `server.xml`.

     ```
     <featureManager>
       <feature>eXtremeScale.rest-1.1</feature>
     </featureManager>
     ```

   - Activez la passerelle REST dans le fichier profil Liberty `server.xml` en utilisant Liberty Profile Developer Tools.
     - Démarrez IBM®WebSphere Application Server Version 8.6 Liberty Profile Developer Tools.
     - Dans l'onglet **Design**, sélectionnez **Feature Manager**. Cliquez sur **Add** dans la section Feature Manager Details. Sélectionnez la fonction **eXtremeScale.rest-1.1** et ajoutez-la.
     - Feature Manager étant sélectionné, cliquez sur **Add** dans la section Feature Manager Details. Sélectionnez la fonction **servlet-3.0** et ajoutez-la.
     - Enregistrez le fichier `server.xml`.
   - Activez la passerelle REST dans WebSphere Application Server.
     - Installez WebSphere eXtreme Scale avec WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
     - Déployez le fichier *was_install_root*`\optionalLibraries\ObjectGrid\restgateway\wxsRESTGateway.war` sur WebSphere Application Server.
2. Configurez la passerelle REST.
   a. Configurez la passerelle REST dans le fichier `server.xml`. Entrez la ligne de code suivante :

      ```
      <xsREST contextRoot=”myContextRoot” remoteDomain=”myDomain”/>
      ```

      Avertissement : Les attributs, `contextRoot` et `remoteDomain`, sont facultatifs. La racine de contenu par défaut est `resources`.

   b. Configurez un serveur eXtreme Scale.

   c. Configurez un serveur de conteneur.

      Les options suivantes sont disponibles pour configurer un service de conteneur :
      - Copiez un fichier valide `objectgrid.xml` (avec ou sans fichier `objectGridDeployment.xml` correspondant) dans le répertoire *rép_base_wlp*`/usr/servers/`*nom_serveur*`/grids`. Ce répertoire `grids` est surveillé par le produit lors de l'exécution. Les modifications des fichiers dans ce répertoire génèrent des événements dans l'environnement d'exécution profil Liberty. Par exemple, lorsqu'un nouveau fichier `objectgrid.xml` ou `objectGridDeployment.xml` ou ces deux nouveaux fichiers sont détectés, un serveur de conteneur est créé. Lorsque l'un de ces fichiers est supprimé, eXtreme Scale arrête ce serveur de conteneur. Lorsque les fichiers sont modifiés, eXtreme Scale arrête et redémarre le conteneur. Plusieurs conteneurs de fragment peuvent exister dans un même serveur eXtreme Scale, ce qui implique que des sous-répertoires existent dans le répertoire `grids`.
3. Démarrez le serveur profil Liberty pour exécuter la passerelle client REST.

Que faire ensuite

Lorsque la passerelle REST est activée, un utilisateur ayant accès au servlet peut accéder aux données dans une grille de données. Par conséquent, vous devez utiliser la sécurité d'application Web dans WebSphere Application Server pour contrôler l'autorisation. Pour plus d'informations sur la sécurisation des applications Web qui utilisent cette passerelle REST, voir Securing web applications dans le centre de documentation de WebSphere Application Server.

Le fichier `wxsRESTGateway.war`, qui contient le fichier `web.xml` pour la configuration de la sécurité, se trouve dans les emplacements suivants en fonction de votre installation :

- *`wlp_install_root`*`/wxs/web/rest`
- *`was_install_root`*`/optionalLibraries/ObjectGrid/restgateway`
- *`wxs_standalone_install_root`*`/ObjectGrid/restgateway`

Maintenant, vous pouvez utiliser le service de données Web dans le profil Liberty pour communiquer avec la grille de données via un identificateur URI. Pour plus d'informations, voir Développement d'applications de grille de données avec la passerelle REST.

Rubrique parent : Configuration des clients Java

.NET

Configuration de WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET à l'aide du fichier de propriétés du client, de la configuration XML côté serveur, ou en remplaçant à l'aide d'un programme certaines propriétés du serveur.

Pourquoi et quand exécuter cette tâche

Lorsque le client appelle la méthode Connect sur la grille de données, la configuration est effectuée à l'aide du fichier de propriétés du client spécifié. Si vous n'avez pas spécifié de fichier de propriétés, le fichier `Client.Net.properties` est utilisé.

Si des modifications sont apportées au fichier de propriétés du client après que le client a appelé la méthode Connect, vous devez redémarrer la connexion du client à la grille de données pour que ces modifications soient prises en compte.

Vous pouvez configurer la configuration dynamique de certaines propriétés dans le fichier `Client.Net.properties`.

La méthode Connect peut émettre des exceptions si le fichier de configuration du client contient des valeurs de propriété erronées, des propriétés mal placées ou d'autres erreurs. WebSphere eXtreme Scale Client for .NET contient également plusieurs valeurs de propriété de configuration XML côté serveur qui peuvent être remplacées par programme à l'aide de l'API Client for .NET.

.NET 2.5+ **Substitutions pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer WebSphere eXtreme Scale Client for .NET en remplaçant certaines propriétés du serveur. Ces propriétés sont définies dans le fichier de configuration `objectgrid.xml`.

.NET 2.5+ **Configuration de WebSphere eXtreme Scale Client for .NET à l'aide d'un programme**
Vous pouvez remplacer les paramètres côté client à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

.NET 2.5+ **Activation de la configuration dynamique de WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer WebSphere eXtreme Scale Client for .NET pour qu'il détecte dynamiquement les modifications apportées aux valeurs des propriétés du fichier de propriétés du client. Il n'est pas nécessaire de redémarrer la connexion à la grille de données pour que ces modifications prennent effet.

Rubrique parent : Configuration des clients

.NET

Substitutions pour WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET en remplaçant certaines propriétés du serveur. Ces propriétés sont définies dans le fichier de configuration `objectgrid.xml`.

Pour remplacer des paramètres sur un client, vous pouvez procéder à une configuration par programmation sur WebSphere eXtreme Scale Client for .NET. Pour plus d'informations sur les substitutions à l'aide d'un programme, voir Configuration de WebSphere eXtreme Scale Client for .NET à l'aide d'un programme.

Vous pouvez remplacer les attributs suivants sur un client :

Attributs BackingMap

- attribut timeToLive
- attribut LockTimeOut

Attributs ObjectGrid

- attribut txTimeout
- attribut txIsolation

Rubrique parent : .NET 2.5+ Configuration de WebSphere eXtreme Scale Client for .NET

.NET

Configuration de WebSphere eXtreme Scale Client for .NET à l'aide d'un programme

Vous pouvez remplacer les paramètres côté client à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

Pourquoi et quand exécuter cette tâche

L'exemple de code ci-dessous crée les mêmes substitutions que celles décrites dans Substitutions pour WebSphere eXtreme Scale Client for .NET.

Procédure

Le code suivant crée une instance ObjectGrid côté client :

```
IGridManager gm = GridManagerFactory.GetGridManager( );
ICatalogDomainInfo cdi =
gm.CatalogDomainManager.CreateCatalogDomainInfo("localhost:2809");
ctx = gm.Connect(cdi, "Sample.Client.Net.properties");
IGrid grid = gm.GetGrid( ctx, "Grid");

//Overriding grid’s txTimeout value
grid.TransactionTimeout = new TimeSpan(0, 0, 30);

IGridMapPessimisticTxObject, Object gridMap;
gridMap = grid.GetGridMapPessimisticTx<Object, Object>("Map1");

//Overriding timeToLive value
gridMap.TimeToLive = new TimeSpan(0, 0, 50);

//Overriding lockTimeout value
gridMap.LockTimeout = new TimeSpan(0, 0, 20);

//Overriding txTimeout value
gridMap.Transaction.TransactionTimeout = new TimeSpan(0, 0, 40);

//Overriding txIsolation value
gridMap.Transaction.TransactionIsolationLevel = TxnIsolationLevel.ReadUncommitted;

//Calling ResetToDefaults(), resets the value of timeToLive, lockTimeout,
//txTimeout, txIsolation back to values when .NET client initialized the grid.

gridMap.ResetToDefaults();
```

Avant de remplacer la valeur **timeToLive** à l'aide d'un programme, définissez la valeur **ttlEvictorType** sur LAST_ACCESS_TIME ou LAST_UPDATE_TIME dans le fichier `objectgrid.xml`. Si WebSphere eXtreme Scale Client for .NET tente de remplacer **timeToLive** à l'aide d'un programme sans définir la valeur **ttlEvictorType**, une exception est émise. Lorsque **txTimeout** a la valeur `TimeSpan.Zero`, le délai d'attente est illimité.

Rubrique parent : .NET 2.5+ Configuration de WebSphere eXtreme Scale Client for .NET

Information associée:

Propriété IGridTransaction.TransactionTimeout
Propriété IGridMapPessimisticAutoTx(Of TKey, TValue).LockTimeout
Propriété IGridMapPessimisticAutoTx(Of TKey, TValue).TimeToLive
Propriété IGridMapPessimisticTx(Of TKey, TValue).LockTimeout
Propriété IGridMapPessimisticTx(Of TKey, TValue).TimeToLive
Propriété IGrid.TransactionTimeout

.NET

Activation de la configuration dynamique de WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET pour qu'il détecte dynamiquement les modifications apportées aux valeurs des propriétés du fichier de propriétés du client. Il n'est pas nécessaire de redémarrer la connexion à la grille de données pour que ces modifications prennent effet.

Pourquoi et quand exécuter cette tâche

Vous pouvez configurer dynamiquement la propriété requestRetryTimeout afin de spécifier la durée (en millisecondes) pendant laquelle le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Toutes les autres propriétés sont en lecture seule. Si vous modifiez la valeur d'une propriété en lecture seule, la modification n'est pas prise en compte et une erreur est consignée dans les fichiers journaux système.

Procédure

1. Dans votre fichier de propriétés du client, affectez la valeur `true` à la propriété enableDynamicConfiguration pour WebSphere eXtreme Scale Client for .NET.

   .NET 2.5+ **enableDynamicConfiguration**

   Lorsque cette propriété a la valeur `true`, les modifications apportées à la propriété requestRetryTimeout du fichier de propriétés du client sont détectées dynamiquement. La nouvelle valeur de cette propriété est immédiatement utilisée pour calculer le nouveau délai d'attente pour les nouvelles tentatives d'exécution de requête.

   Valeur par défaut : `false`

2. Mettez à jour la valeur de la propriété requestRetryTimeout dans votre fichier de propriétés du client.

   Java .NET **requestRetryTimeout**

   Indique pendant combien de temps (en millisecondes) le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Utilisez l'une des valeurs admises suivantes :
   - Une valeur égale à `0` indique que la requête doit échouer immédiatement et ignorer la logique interne régissant les nouvelles tentatives.
   - Une valeur égale à `-1` indique que le délai entre les tentatives d'exécution de la requête n'est pas défini, ce qui signifie que la durée pendant laquelle le système tente d'exécuter la requête dépend du délai d'expiration des transactions. (Valeur par défaut)
   - Une valeur supérieure à `0` indique la valeur du délai d'expiration de la requête en millisecondes. Les exceptions dont la création échoue sont renvoyées. Même lorsque des exceptions, telles que DuplicateException, sont réexécutées, elles sont également renvoyées quand elles échouent. Le délai de transaction reste utilisé comme délai d'attente maximal.

Résultats

Vous pouvez mettre à jour dynamiquement la valeur de la propriété requestRetryTimeout du client sans redémarrer la connexion à la grille de données.

Rubrique parent : .NET 2.5+ Configuration de WebSphere eXtreme Scale Client for .NET

Administration des grilles de données

Vous pouvez utiliser la console, la ligne de commande et l'interface de commande HTTP pour administrer vos grilles de données.

Demande, affichage et invalidation des données
Vous pouvez utiliser les interfaces de requête dans la console de surveillance et dans l'utilitaire **xscmd** pour extraire de petits ensembles de clés et de valeurs à partir d'une mappe et invalider des ensembles de données.

Administration avec l'utilitaire xscmd
Utilisez l'utilitaire **xscmd** pour effectuer des tâches d'administration dans l'environnement.

Administration à l'aide de l'interface de commande HTTP
A l'aide de l'interface de commande HTTP, vous pouvez effectuer des opérations sur votre dispositif, configurer les paramètres du dispositif et administrer des grilles de données, des collectivités et des zones.

Demande, affichage et invalidation des données

Vous pouvez utiliser les interfaces de requête dans la console de surveillance et dans l'utilitaire **xscmd** pour extraire de petits ensembles de clés et de valeurs à partir d'une mappe et invalider des ensembles de données.

Avant de commencer

- Si vous utilisez **xscmd** pour interroger afficher et invalider des données, configurez l'utilitaire **xscmd**. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser la console ou l'utilitaire **xscmd** pour interroger le contenu d'une grille de données. Vous pouvez interroger les données en exécutant une expression régulière sur la clé de données. Vous pouvez ensuite utiliser la même requête pour invalider les données. Pour des exemples d'expressions régulières voir Syntaxe d'expression régulière.

Procédure

- Demandez affichez ou invalidez des données avec la console.
  1. Accédez à la page de requête dans la console. Dans l'interface utilisateur, cliquez sur **Gestion de données** > **Contenu de la grille de données de requête**.
  2. Recherchez ou filtrez les données dans la mappe. Vous pouvez utiliser l'une des options suivantes pour rechercher ou filtrer les données :
     - Tapez une expression régulière dans la zone, puis cliquez sur le bouton **Rechercher** ( ). Une liste de clés correspondant à l'expression régulière s'affiche. La liste des données peut être un sous-ensemble de toutes les données correspondantes.
     - Pour filtrer les résultats en fonction d'un ensemble de partitions, cliquez sur le bouton **Filtrer** ( ). Vous pouvez ensuite taper une expression régulière et choisir une plage de partitions en fonction de laquelle vous voulez filtrer les résultats.
  3. Affichez les valeurs des clés affichées. Sélectionnez **Afficher les valeurs**. Les valeurs s'affichent dans le tableau. Si la valeur est trop longue à afficher, des points de suspension (. . .) tronquent la valeur. Cliquez sur la valeur pour afficher toute la zone. Les valeurs sont renvoyées sous forme de chaînes de texte. Certaines valeurs peuvent ne pas être converties en chaînes explicites et des valeurs hexadécimales s'affichent.

     Important : L'application peut stocker des valeurs d'objet pour lesquelles la classe Java™ n'est pas reconnue par le serveur. Si l'application utilise eXtreme Data Format (XDF), ces valeurs s'affichent. Si XDF n'est pas utilisé et que la classe Java n'est pas reconnue par le serveur, un message indique que la classe de l'objet n'est pas disponible pour le serveur.
  4. Invalidez les données. Lors de l'invalidation des données, celles-ci sont définitivement supprimées de la grille de données.

     Clés sélectionnées

     Vous pouvez sélectionner dans la table des clés à invalider. Vous pouvez alors cliquer individuellement sur les entrées ou cocher la case Tout sélectionner qui permet de sélectionner un maximum de 500 entrées dans la table. Une fois les entrées à supprimer sélectionnées, cliquez sur **Invalider** > **Clés sélectionnées**.

     Toutes les clés correspondant à la requête

     Vous pouvez également invalider toutes les données correspondant à votre expression régulière. Cette option permet de supprimer toutes les données de la grille de données qui correspondent à l'expression régulière, et pas uniquement le nombre maximal de 500 entrées affichées dans la console. Pour invalider des entrées comportant l'expression régulière, cliquez sur **Invalider** > **Toutes les clés correspondant à la requête**.
  5. Supprimez tout le contenu de la mappe. Cliquez sur **Effacer la mappe**. Vous devez confirmer la suppression de toutes les entrées dans la mappe sélectionnée.
- Demandez affichez ou invalidez des données avec l'utilitaire **xscmd**.

  Interrogation de données :

  ```
  xscmd.sh –c findbykey –g <grille_données> -m <mappe>
  -fs <chaîne_recherche> [-fp <id_partition>]
  ```

  Vous devez inclure la grille de données, la mappe et l'expression régulière pour la valeur de la chaîne

de recherche. Vous pouvez également appliquer un filtrage par ID partition. Le résultat renvoie un sous-ensemble de la totalité de la requête.

Invalidation de données :

Incluez l'argument **-inv** dans la commande pour invalider les données sélectionnées par la requête.

```
xscmd –c findbykey –g <grille_données> -m <mappe>
-fs <chaîne_recherche> [-fp <id_partition>] -inv
```

Vous devez inclure la grille de données, la mappe et l'expression régulière pour la valeur de la chaîne de recherche. Vous pouvez également appliquer un filtre par ID partition. Lors de l'exécution de l'invalidation, toutes les valeurs correspondantes sont invalidées, et pas seulement le petit ensemble renvoyé par la requête.

Affichez les valeurs des données demandées :

Incluez l'argument **-rv** dans la commande pour afficher les valeurs des données sélectionnées par la requête.

```
xscmd.sh –c findbykey –g <grille_données> -m <mappe>
-fs <find_string> -rv
```

Vous devez inclure la grille de données, la mappe et l'expression régulière pour la valeur de la chaîne de recherche. Vous pouvez également appliquer un filtre par ID partition. Le résultat renvoie un sous-ensemble de la totalité de la requête et inclut les valeurs de chaque clé.

UNIX Linux **Important :** Si votre expression régulière commence par les caractères `.*`, il se peut que ces caractères ne soient pas traités correctement lors de l'exécution de la commande. Pour résoudre ce problème, mettez en forme votre expression régulière d'une des manières suivantes :

- Placez votre expression régulière entre des apostrophes : `-fs '.*'`
- Utilisez une barre oblique inversée comme caractère d'échappement pour l'astérisque : `-fs .\*`

Exemple :

Dans l'exemple ci-dessous, toutes les entrées de la grille de données `Grid` et de la mappe `Map1` sont recherchées.

```
xscmd -c findbykey -g Grid -m Map1 -fs ".*"
```

La commande renvoie les résultats suivants :

```
3 matching keys were found.

Partition Key
--------- ---
2         keyghi
4         keydef
6         keyabc
```

Rubrique parent : Administration des grilles de données

Tâches associées:
Administration avec l'utilitaire xscmd

Référence associée:
Référence à l'utilitaire xscmd

Administration avec l'utilitaire `xscmd`

Utilisez l'utilitaire **`xscmd`** pour effectuer des tâches d'administration dans l'environnement.

Avant de commencer

- Le dispositif doit être démarré.
- Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client : Vous devez connaître l'adresse IP et le numéro de port d'un serveur de catalogue actif. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres**. Sélectionnez un membre de collectivité. L'adresse IP et le numéro de port du serveur de catalogue s'affichent.
- Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client : Vérifiez que la variable d'environnement *`JAVA_HOME`* est définie pour utiliser l'environnement d'exécution installé avec le produit. Si vous utilisez la version d'évaluation du produit, vous devez définir la variable d'environnement *`JAVA_HOME`*.

Pourquoi et quand exécuter cette tâche

2.5+ Vous pouvez exécuter l'utilitaire **`xscmd`** à partir d'une installation client ou de l'interface de ligne de commande du dispositif. Lorsque vous l'exécutez à partir de l'interface de ligne de commande, il se connecte automatiquement à un serveur de catalogue de la collectivité. Dans ce cas, vous n'avez pas besoin de définir les variables d'environnement et d'importer le fichier de clés certifiées du dispositif. Vous ne pouvez vous connecter qu'à la collectivité locale. Si vous voulez vous connecter à des collectivités éloignées, vous devez opérer à partir d'une installation client ou utiliser l'interface de ligne de commande de l'un des dispositifs de la collectivité éloignée concernée.

Procédure

1. Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client : Téléchargez sur le client le fichier de clés certifiées actif du dispositif. Dans l'interface utilisateur du dispositif, cliquez sur **Collectivité** > **Paramètres** > **TLS (Transport Layer Security)** > **Télécharger le fichier de clés certifiées actif**. Le fichier de clés certifiées par défaut est le fichier `xsatruststore.jks`. Le mot de passe par défaut est `xc10pass`.
2. Facultatif : Si l'authentification de client est activée : Sur l'installation client, ouvrez une fenêtre de ligne de commande. Sur la ligne de commande, définissez les variables d'environnement appropriées.
3. Connectez l'utilitaire **`xscmd`** au dispositif.
   - Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client :

     Dans le répertoire `bin` de l'installation client, exécutez la commande suivante :

     ```
     xscmd.bat|sh -ts xsatruststore.jks -tst jks -tsp xc10pass -user xcadmin -pwd
     xcadmin
     -cep myxc10.mycompany.com  -prot TLS -cxpv IBMJSSE2 -tt TCP/IP [paramètre
     supplémentaires]
     ```

   - 2.5+ Si vous exécutez l'utilitaire **`xscmd`** à partir de l'interface de ligne de commande du dispositif :
     a. Connectez-vous à l'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.
     b. Exécutez l'utilitaire **`xscmd`**:

        La syntaxe générale de cette commande est la suivante :

        ```
        Console> xscmd -c <nom_commande> -opt1 [arg1] -opt2 [arg2] -opt3
        ```

        La commande suivante affiche l'aide du dispositif :

        ```
        Console> xscmd -h
        ```

4. Affichez l'aide des différentes options **`xscmd`**. Si vous exécutez l'utilitaire **`xscmd`** à partir de l'interface de ligne de commande du dispositif, l'extension `.bat|.sh` n'est pas nécessaire.
   - Pour afficher l'aide générale, exécutez la commande suivante :
     - UNIX `./xscmd.sh -h`
     - Windows `xscmd.bat -h`
   - Pour afficher la liste de toutes les commandes, exécutez la commande suivante :
     - UNIX `./xscmd.sh -lc`

        - Windows `xscmd.bat -lc`
    - Pour afficher l'aide d'une commande, exécutez la commande suivante :
        - UNIX `./xscmd.sh -h nom_commande`
        - Windows `xscmd.bat -h nom_commande`
    - Pour afficher une liste des groupes de commandes, exécutez la commande suivante :
        - UNIX `./xscmd.sh -lcg`
        - Windows `xscmd.bat -lcg`
    - Pour afficher la liste des commandes d'un groupe de commandes, exécutez la commande suivante :
        - UNIX `./xscmd.sh -lc nom_groupe_commandes`
        - Windows `xscmd.bat -lc nom_groupe_commandes`
5. Exécutez les commandes de connexion à de serveurs de catalogue spécifiques. Vous devez fournir une ou plusieurs combinaisons de port et d'adresse IP de serveur de catalogue pour extraire des informations relatives aux grilles de données exécutées sur le dispositif.Lorsque vous utilisez l'interface de ligne de commande du dispositif, vous ne pouvez vous connecter qu'à la collectivité locale. Si vous voulez vous connecter à des collectivités éloignées, vous devez opérer à partir d'une installation client ou utiliser l'interface de ligne de commande de l'un des dispositifs de la collectivité éloignée concernée.
    - Fournissez la liste des serveurs de catalogue auxquels vous voulez vous connecter :
        - UNIX `./xscmd.sh -c <nom_commande> -cep nom_hôte:port(,nom_hôte:port)`
        - Windows `xscmd.bat -c <nom_commande> -cep nom_hôte:port(,nom_hôte:port)`

      Dans les commandes précédentes, *nom_commande* désigne le nom de la commande que vous exécutez. La valeur *nom_hôte:port* représente le nom d'hôte du serveur de catalogue et le port d'écoute.

ATTENTION :

N'utilisez pas les commandes suivantes dans un environnement WebSphere DataPower XC10 Appliance :

- **-c releaseShard**
- **-c reserveShard**
- **-c swapShardWithPrimary**
- **-c suspendBalancing**
- **-c resumeBalancing**
- **-c teardown**
- **-c triggerPlacement**
- **-c enableForPlacement**

6. Facultatif : Définissez une valeur de délai d'attente lorsque vous exécutez vos commandes. Vous pouvez utiliser l'option **-to** ou **--timeout** comme paramètre global dans n'importe quelle commande. Cette valeur définit le nombre de secondes avant l'expiration du délai d'attente lorsque vous vous connectez à des serveurs de catalogue dans la commande. Si vous vous connectez à un serveur de catalogue qui risque d'être indisponible à la suite de l'expiration d'un délai d'attente du système d'exploitation ou du réseau, cette option peut être utile pour réduire l'attente.

   Le délai d'attente par défaut est de 30 secondes.

Configuration des profils de sécurité pour l'utilitaire xscmd
En créant un profil de sécurité, vous pouvez utiliser les paramètres de sécurité enregistrés pour utiliser l'utilitaire **xscmd** avec des environnements sécurisés.

Rubrique parent : Administration des grilles de données

Tâches associées:
Demande, affichage et invalidation des données

Référence associée:
Référence à l'utilitaire xscmd

Configuration des profils de sécurité pour l'utilitaire `xscmd`

En créant un profil de sécurité, vous pouvez utiliser les paramètres de sécurité enregistrés pour utiliser l'utilitaire **`xscmd`** avec des environnements sécurisés.

Avant de commencer

Pour plus d'informations sur la configuration de l'utilitaire **`xscmd`**, voir Administration avec l'utilitaire xscmd.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser le paramètre **`-ssp`** *`nom_profil`* ou **`--saveSecProfile`** *`nom_profil`* avec le reste de la commande **`xscmd`** pour enregistrer un profil de sécurité. Le profil peut contenir des paramètres pour les noms d'utilisateur et les mots des générateurs de données d'identification, des fichiers de clés, des fichiers de clés certifiées et des types de transport.

Le groupe de commandes **`ProfileManagement`** dans l'utilitaire **`xscmd`** contient des commandes de gestion de vos profils de sécurité.

Procédure

- Enregistrez un profil de sécurité.

  Pour enregistrer un profil de sécurité, utilisez le paramètre **`-ssp`** *`nom_profil`* ou **`--saveSecProfile`** *`nom_profil`* avec le reste de la commande. L'ajout de ce paramètre à la commande enregistre les paramètres suivants :

  ```
  -al,--alias <alias>
  -arc,--authRetryCount <integer>
  -ca,--credAuth <support>
  -cgc,--credGenClass <className>
  -cgp,--credGenProps <property>
  -cxpv,--contextProvider <provider>
  -ks,--keyStore <filePath>
  -ksp,--keyStorePassword <password>
  -kst,--keyStoreType <type>
  -prot,--protocol <protocol>
  -pwd,--password <password>
  -ts,--trustStore <filePath>
  -tsp,--trustStorePassword <password>
  -tst,--trustStoreType <type>
  -tt,--transportType <type>
  -user,--username <username>
  ```

  Les profils de sécurité sont enregistrés dans le répertoire *`rép_base_utilisateur`*`\.scmd\profiles\security\<`*`nom_profil`*`>.properties`.

  Important : N'incluez pas l'extension de nom de fichier `.properties` dans le paramètre *`nom_profil`*. Cette extension est automatiquement ajoutée au nom de fichier.

- Utilisez un profil de sécurité enregistré.

  Pour utiliser un profil de sécurité enregistré, ajoutez le paramètre **`-sp`** *`nom_profil`* ou **`--securityProfile`** *`nom_profil`* à la commande que vous exécutez.

  Exemple de commande : `xscmd -c listHosts -cep myhost.mycompany.com -sp myprofile`

- Listez les commandes dans le groupe de commandes **`ProfileManagement`**.

  Exécutez la commande **`xscmd -lc ProfileManagement`**.

- Listez les profils de sécurité existants.

  Exécutez la commande **`xscmd -c listProfiles -v`**.

- Affichez les paramètres enregistrés dans un profil de sécurité.

  Exécutez la commande **`xscmd -c showProfile -pn`** *`nom_profil`*.

- Supprimez un profil de sécurité existant.

  Exécutez la commande **`xscmd -c RemoveProfile -pn`** *`nom_profil`*.

Rubrique parent : Administration avec l'utilitaire xscmd

Référence associée:
Référence à l'utilitaire xscmd

Administration à l'aide de l'interface de commande HTTP

A l'aide de l'interface de commande HTTP, vous pouvez effectuer des opérations sur votre dispositif, configurer les paramètres du dispositif et administrer des grilles de données, des collectivités et des zones.

Pourquoi et quand exécuter cette tâche

L'interface de commande HTTP permet d'effectuer des opérations avec les instructions JSON HTTP POST. Vous pouvez regrouper ces instructions dans des scripts pour automatiser les tâches de configuration et d'administration.

Procédure

1. Visualisez les commandes disponibles pour l'interface de commande HTTP dans l'interface utilisateur.
   Pour visualiser une table de toutes les commandes disponibles dans l'interface utilisateur, cliquez sur **(Aide)** > **Interface de commande HTTP - Aide**. Cliquez sur un nom de commande pour afficher la **Syntaxe détaillée** et un **Exemple de soumission JSON**.

   Pour afficher la liste de toutes les commandes dans le centre de documentation, voir Guide de référence de l'interface de commande HTTP.
2. Créez une instruction JSON pour l'opération que vous souhaitez effectuer. Cette instruction doit contenir :
   - La commande que vous souhaitez exécuter.
   - Les paramètres appropriés pour la commande.

   Utilisez la commande suivante dans l'outil **cURL** :

   ```
   curl -v -k -u UTIL_ADMIN_XC:MDP_ADMIN_XC -H "Content-Type: application/json" --data-binary 'INSTRUCTION_JSON_INTERFACE_HTTP'
   https://NOM_HOTE_XC10/resources/appTaskInterface
   ```

   Définissez les variables suivantes :

   UTIL_ADMIN_XC:MDP_ADMIN_XC

   Indique le nom d'utilisateur et le mot de passe de l'administrateur WebSphere DataPower XC10 Appliance.

   INSTRUCTION_JSON_INTERFACE_HTTP

   Indique l'une des instructions JSON possibles soumises à l'interface de commande HTTP. Vous pouvez copier l'instruction **Exemple de soumission JSON** à partir de **Interface de commande HTTP - Aide** pour la commande que vous voulez exécuter. La commande doit être placées entre apostrophes (').

   NOM_HOTE_XC10

   Indique le nom d'hôte complet ou l'adresse IP de WebSphere DataPower XC10 Appliance.

   Par exemple, vous pouvez copier l'exemple de soumission JSON de la commande ViewAllUsers à partir de l'aide de l'interface de commande HTTP. Puis vous pouvez exécuter la commande suivante :

   ```
   curl -v -k -u xcadmin:xcadmin -H "Content-Type: application/json" --data-binary
   '{"task":{"stopOnTaskFailure":"true","command":"ViewAllUsers"}}'
   https://myXC10.mycompany.com/resources/appTaskInterface
   ```
3. Exécutez la commande et consultez sa sortie.

Rubrique parent : Administration des grilles de données

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Information associée:
Outils cURL et libcurl
Guide de référence de l'interface de commande HTTP

Développement d'applications pour accéder à des grilles de données simples

Vous pouvez utiliser deux options différentes pour créer des mappes et lire, écrire et mettre à jour des grilles de données simples. Vous pouvez écrire une application client Java qui utilise l'API ObjectMap. Vous pouvez aussi utiliser la passerelle REST pour développer une application non-Java pour accéder à la grille de données simple.

Avant de commencer

Vous devez disposer d'une grille de données simple existante. Pour plus d'informations sur la création d'une grille de données simple, voir Création de grilles de données simples.

Si vous utilisez une mémoire cache dynamique ou une grille de données de session, vous pouvez utiliser vos applications existantes sans modification.

Développement d'applications de grilles de données avec des API Java
Vous pouvez vous connecter au serveur de catalogue, obtenir une instance ObjectGrid et utiliser l'API ObjectMap.

Développement d'applications de grille de données avec la passerelle REST
Vous pouvez utiliser la passerelle REST (Representational State Transfer) pour accéder à des grilles de données simples qui sont hébergées par une collectivité. Cette passerelle REST est pratique lorsque vous devez accéder à une grille de données à partir d'environnements non Java.

.NET 2.5+ **Développement d'applications de grille de données avec les API .NET**
Vous pouvez développer des applications Microsoft .NET qui utilisent la même grille de données que vos applications Java™.

Développement d'applications de grilles de données avec des API Java

Vous pouvez vous connecter au serveur de catalogue, obtenir une instance ObjectGrid et utiliser l'API ObjectMap.

Avant de commencer

Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.

Visite virtuelle d'application de grille de données simple

1. **Connectez-vous au service de catalogue via une instance ClientClusterContext.**

   Le serveur de catalogue à spécifier est affiché dans la page d'interface utilisateur de la grille de données simple que vous avez créée. Cliquez sur **Grille de données** > **Grille de données simple** > ***my_simple_data_grid*** et utilisez les valeurs dans la zone **Services de catalogue**.

   Pour établir la connexion au serveur de catalogues, utilisez la méthode connect de l'API ObjectGridManager. La méthode connect utilisée requiert seulement un noeud final de serveur de catalogue au format *nom_hôte*:*port*. Vous pouvez indiquer plusieurs noeuds finaux de serveur de catalogue en séparant les valeurs *hostname*:*port* par une virgule. Le fragment de code suivant montre comment se connecter à un serveur de catalogue et obtenir une instance ClientClusterContext :

   ```
   ClientClusterContext ccc =
   ObjectGridManagerFactory.getObjectGridManager().connect("myXC10.myhost.com:2809",
   null, null);
   ```

   La méthode connect tente de se connecter à chaque dispositif de la liste jusqu'à ce qu'elle puisse établir une connexion. Un basculement automatique est effectué si l'un des autres dispositifs ne répond pas.

   Si les connexions aux serveurs de catalogue aboutissent, la méthode connect retourne une instance ClientClusterContext. L'instance ClientClusterContext est requise pour obtenir l'ObjectGrid à partir de l'API ObjectGridManager.
2. **Obtenez une instance ObjectGrid.**

   Pour obtenir une instance ObjectGrid, utilisez la méthode getObjectGrid de l'API ObjectGridManager. La méthode getObjectGrid requiert l'instance ClientClusterContext et le nom de l'instance de grille de données. L'instance ClientClusterContext est obtenue pendant la connexion au serveur de catalogue. Le nom de l'instance de grille de données est le nom de la grille de données simple que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir la grille de données en appelant la méthode getObjectGrid de l'API ObjectGridManager.

   ```
   ObjectGrid grid = ObjectGridManagerFactory.getObjectGridManager().getObjectGrid(ccc,
   “my_simple_data_grid”);
   ```

3. **Définissez les droit d'accès de sécurité nécessaires.**

   Créez une configuration de sécurité du client avec un nom d'utilisateur et un mot de passe que vous fournissez à l'application. Le nom d'utilisateur et le mot de passe que vous utilisez doivent avoir l'autorisation d'accéder à la grille de données sur le dispositif. Pour plus d'informations sur la création d'un utilisateur autorisé, voir Gestion des utilisateurs et des groupes.

   ```
                  // Creates a ClientSecurityConfiguration object using the specified
   file
                  ClientSecurityConfiguration clientSC =
   ClientSecurityConfigurationFactory.getClientSecurityConfiguration();
                  clientSC.setSecurityEnabled(true);
                  // Creates a CredentialGenerator using the passed-in user and
   password.
                  CredentialGenerator credGen = new
   UserPasswordCredentialGenerator(username,password);
                  clientSC.setCredentialGenerator(credGen);
                  return clientSC;
   ```

4. **Obtenez une instance Session.**

   Vous pouvez obtenir une session de l'instance ObjectGrid obtenue. Une instance Session est

indispensable pour obtenir l'instance ObjectMap et pour effectuer une démarcation de transaction. Le fragment de code suivant montre comment obtenir une instance Session en appelant la méthode getSession de l'API ObjectGrid.

```
Session sess = grid.getSession();
```

5. **Obtenez une instance ObjectMap.**

   Après avoir obtenu une instance Session, vous pouvez obtenir une instance ObjectMap à partir d'une instance Session en appelant la méthode getMap de l'API Session. Le nom d'instance de mappe que vous envoyez à la méthode getMap porte le nom de la grille de données que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir ObjectMap en appelant la méthode getMap de l'API Session.

   ```
   ObjectMap map1 = sess.getMap("my_simple_data_grid");
   ```

   L'exemple précédent utilise l'instance de mappe par défaut qui est nommé d'après la grille de données. Vous pouvez également indiquer un nouveau nom de mappe, comme dans les exemples suivants :

   ```
   ObjectMap map2 = sess.getMap("my_simple_data_grid.CT.P");
   ObjectMap map3 = sess.getMap("my_new_map.NONE");
   ```

   La mappe `my_simple_data_grid.CT.P` est une mappe qui utilise l'expulsion en fonction de l'heure de création et le verrouillage pessimiste. La mappe `my_new_map.NONE` ne dispose pas de paramètres d'expulsion ou de verrouillage. Pour plus d'informations, voir Options de configuration de mappe dynamique.
6. **Utilisez les méthodes ObjectMap.**

   Une fois une instance ObjectMap obtenue, vous pouvez utiliser l'API ObjectMap. N'oubliez pas que l'interface ObjectMap est une mappe transactionnelle et qu'elle requiert une démarcation de transaction à l'aide des méthodes begin et commit de l'API Session. Faute de démarcation de transaction explicite, les opérations ObjectMap s'exécutent avec des transactions de validation automatique.

   Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une transaction de validation automatique.

   ```
   map1.insert(key1, value1);
   ```

   La clé que vous utilisez peut avoir n'importe quel type Java existant, tel que java.lang.String ou Entier. Les valeurs peuvent correspondre à n'importe quel type d'objet sérialisable.

   Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une démarcation de transaction explicite.

   ```
   sess.begin();
   map1.insert(key1, value1);
   sess.commit();
   ```

Accès à la documentation des API Java

Vous pouvez accéder à la documentation des API Java™ pour WebSphere DataPower XC10 Appliance en téléchargeant un fichier archive zip, en intégrant la documentation des API Dans l'environnement de développement ou en l'affichant dans le centre de documentation.

Java Exemple : application grille de données simple

Cet exemple utilise l'API ObjectMap pour effectuer des opérations de création, de récupération, de mise à jour et de suppression simples sur la grille de données.

Création de mappes dynamiques avec les API Java

Vous pouvez créer des mappes dynamiques avec des API Java une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

2.5+ Programmation de transactions dans des applications Java

Lorsque vous écrivez une application Java qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

Plug-in d'indexation des données

Selon le type d'index que vous voulez générer, WebSphere eXtreme Scale Client fournit des plug-in intégrés que vous pouvez ajouter à la mappe de sauvegarde pour générer un index.

2.5+ **Notification aux clients des mises à jour de mappe en utilisant l'interrogation continue**
Vous pouvez être notifié dans votre machine JVM (Java virtual machine) client de l'insertion ou de la mise à jour d'objets ou d'entrées dans la grille de données.

Rubrique parent : Développement d'applications pour accéder à des grilles de données simples

Tâches associées:
Création de grilles de données simples

Référence associée:
Exemple : application grille de données simple
Fichier de propriétés du client

Information associée:
Spécification de l'API client

Java

Accès à la documentation des API Java

Vous pouvez accéder à la documentation des API Java™ pour WebSphere DataPower XC10 Appliance en téléchargeant un fichier archive zip, en intégrant la documentation des API Dans l'environnement de développement ou en l'affichant dans le centre de documentation.

Pourquoi et quand exécuter cette tâche

Vous pouvez accéder à la documentation des API Java dans l'un des emplacements suivants :

Centre de documentation

L'utilisation de la documentation d'API du centre de documentation sert à effectuer des recherches conjointement avec le reste des informations de produit WebSphere DataPower XC10 Appliance.

Archive de fichier zip

Vous pouvez télécharger ce fichier pour chaque édition. Vous pouvez ensuite comparer les outils pour déterminer quelles API ont changé d'une édition à l'autre. Vous pouvez également lier directement le fichier compressé dans vos projets Eclipse lors de la compilation au niveau du fichier objectgrid.jar. Ce lien permet d'intégrer la documentation d'API à l'environnement de développement intégré (IDE).

Format en ligne

Le format en ligne est un exemplaire publié de la documentation des API sur le site Web d'IBM®. Vous pouvez vous connecter directement à cette URL dans Eclipse. Le lien de la version en cours est toujours mis à niveau vers la dernière version, de sorte que vous pouvez automatiquement visualiser les corrections et les modifications de la documentation.

Procédure

- Consultez la documentation d'API dans le centre de documentation. Pour plus d'informations, voir Documentation d'API.
- Téléchargez une archive zip de la documentation d'API.

  Pour télécharger la documentation d'API afin de la consulter hors ligne, vous pouvez télécharger un fichier zip correspondant à l'édition appropriée, à partir de la page suivante : WebSphere DataPower XC10 Appliancewiki : Documentation d'API.
- Consultez le format en ligne de la documentation d'API. Vous pouvez associer un signet à un lien qui est est toujours mis à niveau vers la dernière version, ou vous pouvez vous connecter à une version spécifique. Pour obtenir une liste de liens, voir WebSphere DataPower XC10 Appliance wiki : Documentation d'API.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Java

Exemple : application grille de données simple

Cet exemple utilise l'API ObjectMap pour effectuer des opérations de création, de récupération, de mise à jour et de suppression simples sur la grille de données.

Exemple d'application

SimpleGrid.java

```
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.HashSet;
import java.util.BitSet;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.PrintWriter;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.Serializable;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import com.ibm.websphere.objectgrid.ClientClusterContext;
import com.ibm.websphere.objectgrid.ConnectException;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.ObjectGridException;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridRuntimeException;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.config.BackingMapConfiguration;
import com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory;
import com.ibm.websphere.objectgrid.config.ObjectGridConfiguration;
import com.ibm.websphere.objectgrid.plugins.TransactionCallbackException;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory;
import com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator;
import
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator;
import com.ibm.websphere.objectgrid.server.CatalogServerProperties;

public class SimpleGrid {

        static String gridName = "test";
        static String mapName = gridName;
        static String username="xcadmin";
        static String password="xcadmin";
        static String hostName = "localhost";
        static ObjectGrid clientGrid=null;
        static ConcurrentLinkedQueue<Session> sessions = new
ConcurrentLinkedQueue<Session>();

        static synchronized public ObjectGrid getObjectGrid() {
                if (clientGrid == null) {
                        ClientClusterContext ccc = null;
                        try {
                           new
```

```
java.io.File(System.getProperty("java.io.tmpdir")).mkdirs();
                        } catch (Throwable t) {
                                t.printStackTrace();
                        }
                        ObjectGridManager ogm =
ObjectGridManagerFactory.getObjectGridManager();
                        ClientSecurityConfiguration clientSC = getAdminClientConfig();
                        List<ObjectGridConfiguration> ogConfigs=new
ArrayList<ObjectGridConfiguration>();
                        ObjectGridConfiguration lclGridConfig =
ObjectGridConfigFactory.createObjectGridConfiguration(gridName);
                        BackingMapConfiguration bmc =
ObjectGridConfigFactory.createBackingMapConfiguration(mapName);
                        bmc.setNumberOfBuckets(0);
                        lclGridConfig.addBackingMapConfiguration(bmc);
                        ogConfigs.add(lclGridConfig);
                        try {
                                ccc = ogm.connect(hostName+":2809", clientSC, null);
                        } catch (Throwable e) {
                                e.printStackTrace();
                        }
                        if (ccc != null) {
                           HashMap<String,List<ObjectGridConfiguration>> overrideMap = new
HashMap<String,List<ObjectGridConfiguration>>();
                           overrideMap.put(ccc.getClusterName(),ogConfigs);
                           ogm.setOverrideObjectGridConfigurations(overrideMap);
                           try {
                                clientGrid = ogm.getObjectGrid(ccc, gridName);
                           } catch (ObjectGridRuntimeException ogre) {
                                ogre.printStackTrace();
                           }
                        }
                }
                return clientGrid;
        }

        static public Session getSession() throws TransactionCallbackException,
                        ObjectGridException {
                Session session = sessions.poll();
                if (session == null && getObjectGrid()!=null) {
                        session = getObjectGrid().getSession();
                }
                if (session == null)
                        throw new IllegalStateException("unable to initialize connection
to objectgrid");
                return session;
        }

        static public void putSession(Session session) {
                if (session.isTransactionActive()) {
                        try {
                                session.rollback();
                        } catch (Exception e) {
                        }
                sessions.add(session);
        }

        public static ClientSecurityConfiguration getAdminClientConfig() {
                // Creates a ClientSecurityConfiguration object using the specified file
                ClientSecurityConfiguration clientSC =
ClientSecurityConfigurationFactory.getClientSecurityConfiguration();
                clientSC.setSecurityEnabled(true);
                // Creates a CredentialGenerator using the passed-in user and password.
                CredentialGenerator credGen = new
UserPasswordCredentialGenerator(username,password);
                clientSC.setCredentialGenerator(credGen);
                return clientSC;
```

```
        }

    public static void main(String args[]) throws Exception {
    for (int i=0;i<args.length;i++) {
         if (args[i].startsWith("-username:")) {
            username = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-password:")) {
            password = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-gridname:")) {
            gridName = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-mapname:")) {
            mapName = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-hostname:")) {
            hostName = args[i].substring(args[i].indexOf(":") + 1);
         } else {
            System.out.println("usage: SimpleGrid [optional args]");
            System.out.println("       -username:<nomutilisateur>");
            System.out.println("       -password:<motdepasse>");
            System.out.println("       -gridname:<nomgrille>");
            System.out.println("       -mapname:<nommappe>");
            System.out.println("       -hostname:<nomhôte>");
            System.exit(1);
         }
    }
    System.out.println("-----------------------------------------------------------------");
    System.out.println("Test de grille simple");
    System.out.println("-----------------------------------------------------------------");
    System.out.println("username     : " + username);
    System.out.println("password     : " + password);
    System.out.println("gridname     : " + gridName);
    System.out.println("mapname      : " + mapName);
    System.out.println("hostname     : " + hostName);
    System.out.println("-----------------------------------------------------------------");

    if (getObjectGrid() == null) {
        System.out.println("ERREUR : impossible de se connecter à objectgrid à "       +
hostName);
        System.exit(1);
    }
             Session session = getSession();
             ObjectMap map=session.getMap(mapName);
             session.begin();
             Object data = map.get("TestKey");
             if ( data != null )
                 map.remove("TestKey");
             map.insert("TestKey", "TestValue");
             session.commit();
             putSession(session);
     }
}
```

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Concepts associés:
Développement d'applications de grilles de données avec des API Java

Tâches associées:
Création de grilles de données simples

Information associée:
Spécification de l'API client

Java

Création de mappes dynamiques avec les API Java

Vous pouvez créer des mappes dynamiques avec des API Java une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Avant de commencer

Choisissez les options de configuration que vous souhaitez utiliser dans votre mappe dynamique. Pour plus d'informations, voir Options de configuration de mappe dynamique.

Procédure

Appelez la méthode Session.getMap(String).

Si vous transmettez une chaîne qui correspond à l'expression régulière d'un des modèles prédéfinis, la mappe appropriée est créée.

```
ObjectMap map2 = sess.getMap("my_simple_data_grid.CT.P");
ObjectMap map3 = sess.getMap("my_new_map.NONE");
```

La mappe my_simple_data_grid.CT.P est une mappe qui utilise l'expulsion en fonction de l'heure de création, le verrouillage pessimiste et l'invalidation du cache local. La mappe my_new_map.NONE ne dispose pas de paramètres d'expulsion ou d'invalidation du cache local.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données
Développement d'applications de grilles de données avec des API Java

Référence associée:
Options de configuration de mappe dynamique
Exemple de passerelle REST : Création de mappes dynamiques

Programmation de transactions dans des applications Java

Lorsque vous écrivez une application Java™ qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

Java 2.5+ **Interaction avec les données dans une transaction pour les applications Java**
Utilisez des sessions pour interagir avec les données à l'aide des opérations insert et update.

2.5+ **Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction**
Si vous répartissez les données dans plusieurs partitions dans la grille de données, vous pouvez lire et mettre à jour plusieurs partitions dans une seule transaction. Ce type de transaction, appelée transaction multipartition, utilise le protocole de validation en deux phases pour coordonner et restaurer la transaction en cas d'échec.

2.5+ **Utilisation du verrouillage**
Les verrous comportent des cycles de vie et leurs différents types sont compatibles entre eux selon plusieurs critères. Les verrous doivent être traités dans un ordre approprié pour éviter les situations d'interblocage.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Java

Interaction avec les données dans une transaction pour les applications Java

Utilisez des sessions pour interagir avec les données à l'aide des opérations insert et update.

Pourquoi et quand exécuter cette tâche

L'interface ObjectMap offre les opérations put, get et remove habituelles. Nous vous recommandons toutefois d'utiliser des noms d'opérations plus précis tels que get, getForUpdate, insert, update et remove. Ces méthodes permettent d'exécuter des opérations de façon plus précise que les API Map habituelles.

Java **Remarque :** Les méthodes upsert et upsertAll remplacent les méthodes ObjectMap put et putAll. Utilisez la méthode upsert pour indiquer à la mappe de sauvegarde qu'une entrée de la grille de données doit placer la clé et la valeur dans la grille. La mappe de sauvegarde exécute une insertion ou une mise à jour pour placer la valeur dans la grille .

Vous pouvez également utiliser la prise en charge de l'indexation, qui est flexible.

Procédure

- Insertion de données.

  Dès que vous avez obtenu une session, vous pouvez utiliser le fragment de code suivant pour insérer des données à l'aide de l'API Map :

  ```
  Session session = ...;
  ObjectMap personMap = session.getMap("PERSON");
  session.begin();
  Person p = new Person();
  p.name = "John Doe";
  personMap.insert(p.name, p);
  session.commit();
  ```

- Mise à jour de données.

  Vous pouvez utiliser le fragment de code suivant pour mettre à jour des données à l'aide de l'API Map.

  ```
  session.begin();
  Person p = (Person)personMap.getForUpdate("John Doe");
  p.name = "John Doe";
  p.age = 30;
  personMap.update(p.name, p);
  session.commit();
  ```

  L'application utilise normalement la méthode getForUpdate plutôt que la simple méthode get pour verrouiller un enregistrement. La méthode de mise à jour doit être appelée pour fournir la valeur mise à jour à la mappe. Si tel n'est pas le cas, la mappe n'est pas modifiée.

- 2.5+ Insertion de données avec le protocole de validation en deux phases en appelant la méthode `session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE); session.begin();`. Le fragment de code suivant montre comment créer, extraire, mettre à jour et supprimer des données dans une grille avec un protocole de validation en deux phases .

  ```
  Session session = og.getSession();
  Objectmap map1 = session.getMap("Map1");
  Objectmap map2 = session.getMap("Map2");
  Objectmap map3 = session.getMap("Map3");
  session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE);
  session.begin();
  map1.insert("randKey345", "HelloMap1");
  map2.insert("randKey58901", "HelloMap2");
  map3.insert("randKey58", "HelloMap3");
  session.commit();
  ```

Rubrique parent : 2.5+ Programmation de transactions dans des applications Java

Concepts associés:
Accès aux données et transactions

Java 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Java 2.5+

Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Si vous répartissez les données dans plusieurs partitions dans la grille de données, vous pouvez lire et mettre à jour plusieurs partitions dans une seule transaction. Ce type de transaction, appelée transaction multipartition, utilise le protocole de validation en deux phases pour coordonner et restaurer la transaction en cas d'échec.

2.5+ **Validation en deux phases et reprise sur incident**
Le protocole de validation en deux phases coordonne toutes les partitions qui participent à une transaction répartie, en déterminant si la transaction doit être validée ou annulée.

2.5+ **Développement d'applications pour écrire dans des transactions multipartitions pour WebSphere eXtreme Scale dans un environnement autonome**
Vous pouvez écrire une application pour une grille de données répartie avec plusieurs partitions dans l'environnement WebSphere eXtreme Scale autonome.

2.5+ **Développement de composants client eXtreme Scale en vue d'utiliser des transactions**
L'adaptateur de ressources WebSphere eXtreme Scale fournit une gestion des connexions client et une prise en charge des transactions locales. Ces prises en charge permettent aux applications Java Platform, Enterprise Edition (Java EE) de rechercher les connexions client eXtreme Scale et de démarquer les transactions locales à l'aide des transactions locales Java EE ou des API eXtreme Scale.

Rubrique parent : 2.5+ Programmation de transactions dans des applications Java

Tâches associées:
Java Interaction avec les données dans une transaction pour les applications Java

Java

Développement d'applications pour écrire dans des transactions multipartitions pour WebSphere eXtreme Scale dans un environnement autonome

2.5+ Vous pouvez écrire une application pour une grille de données répartie avec plusieurs partitions dans l'environnement WebSphere eXtreme Scale autonome.

Avant de commencer

Activez le protocole eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).

Restriction : Tenez compte des restrictions ci-dessous avant de développer des applications destinées à écrire dans des transactions multipartitions.

- Vous ne pouvez pas utiliser la réplication multimaître avec des transactions qui écrivent dans plusieurs partitions.
- Vous ne pouvez pas utiliser plusieurs partitions dans un client WebSphere eXtreme Scale dans une environnement .NET.
- Les mappes de sauvegarde configurées avec un plug-in de chargeur peuvent lire, mais pas écrire dans la mappe d'une transaction multipartition.
- Les mappes de sauvegarde qui utilisent une stratégie de verrouillage NONE ne peuvent pas participer aux transactions multipartitions.

Pourquoi et quand exécuter cette tâche

Utilisez l'API de session TxCommitProtocol définie pour activer le support de transaction multipartition pour WebSphere eXtreme Scale dans un environnement autonome. La nouvelle API fournit les deux options suivantes :

- TxCommitProtocol.ONEPHASE : constante du protocole de validation de transaction qui indique que la transaction doit être validée avec la validation en une étape par défaut. Avec cette option, une transaction peut lire plusieurs partitions, mais elle ne peut écrire que dans une seule partition. Une exception TransactionException se produit si la transaction écrit dans plusieurs partitions.
- TxCommitProtocol.TWOPHASE : constante du protocole de validation de transaction qui indique que la transaction doit être validée avec la validation en une ou deux phases. Si la transaction écrit dans une seule partition, le protocole de validation en une phase est utilisé. Dans le cas contraire, le protocole en deux phases est utilisé pour valider la transaction, impliquant des opérations d'écriture sur plusieurs partitions.

Vous pouvez configurer le support multitransaction pour WebSphere eXtreme Scale dans WebSphere Application Server. Pour plus d'informations, voir Développement de composants client eXtreme Scale en vue d'utiliser des transactions.

Procédure

1. Connectez-vous à la grille de données. Pour plus d'informations, voir Développement d'applications de grilles de données avec des API Java.
2. Obtenez une instance de session de la grille de données avec la méthode ObjectGrid.getSession. Pour plus d'informations, voir Développement d'applications de grilles de données avec des API Java.
3. Activez un protocole de validation en deux phases en définissant le fragment de code suivant : ²`session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE); session.begin();` Le fragment de code suivant montre comment créer, extraire, mettre à jour et supprimer des opérations dans une grille avec un protocole de validation en deux phases :

```
Session session = og.getSession();
Objectmap map1 = session.getMap("Map1");
Objectmap map2 = session.getMap("Map2");
Objectmap map3 = session.getMap("Map3");
session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE);
session.begin();
map1.insert("randKey345", "HelloMap1");
map2.insert("randKey58901", "HelloMap2");
map3.insert("randKey58", "HelloMap3");
session.commit();
```

Que faire ensuite

Vous pouvez activer la fonction de trace sur les transactions multipartitions. Pour plus d'informations, voir Analyse des données de journal et de trace.

Rubrique parent : 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Concepts associés:
Java Stratégies de verrouillage
Java Accès aux données et transactions

Java

Développement de composants client eXtreme Scale en vue d'utiliser des transactions

L'adaptateur de ressources WebSphere eXtreme Scale fournit une gestion des connexions client et une prise en charge des transactions locales. Ces prises en charge permettent aux applications Java™ Platform, Enterprise Edition (Java EE) de rechercher les connexions client eXtreme Scale et de démarquer les transactions locales à l'aide des transactions locales Java EE ou des API eXtreme Scale.

Avant de commencer

Créez une référence de ressource de fabrique de connexions eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser les API d'accès aux données eXtreme Scale de plusieurs façons. Dans tous les cas, la fabrique de connexions eXtreme Scale doit être injectée dans le composant d'application ou recherchée dans l'interface JNDI (Java Naming Directory Interface). Une fois la fabrique de connexions recherchée, vous pouvez démarquer les transactions et créer des connexions permettant d'accéder aux API eXtreme Scale.

Vous pouvez aussi éventuellement transtyper l'instance javax.resource.cci.ConnectionFactory en une fabrique com.ibm.websphere.xs.ra.XSConnectionFactory qui fournit des options supplémentaires pour l'extraction des descripteurs de connexion. Les descripteurs de connexions générés doivent être transtypés vers l'interface com.ibm.websphere.xs.ra.XSConnection, qui fournit la méthode getSession. La méthode getSession renvoie un descripteur d'objet com.ibm.websphere.objectgrid.Session qui permet aux applications d'utiliser n'importe laquelle des API d'accès aux données eXtreme Scale, telles que les API ObjectMap et EntityManager.

Le descripteur de session et les objets dérivés sont valides pendant toute la durée de vie du descripteur XSConnection.

Les procédures suivantes peuvent être utilisées pour démarquer les transactions eXtreme Scale. Vous ne pouvez pas mélanger ces procédures. Par exemple, vous ne pouvez pas mélanger une démarcation de transaction globale et une démarcation de transaction locale dans le même contexte de composant d'application.

Procédure

- Utilisez des transactions locales à validation automatique. Suivez les étapes ci-dessous pour utiliser automatiquement les opérations d'accès aux données à validation ou les opérations qui ne prennent pas en charge une transaction active :
    1. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection en dehors du contexte d'une transaction globale.
    2. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session pour interagir avec la grille de données.
    3. Appelez toute opération d'accès aux données prenant en charge les transactions à validation automatique.
    4. Fermez la connexion.
- Utilisez une session ObjectGrid pour démarquer une transaction locale. Suivez les étapes ci-dessous pour démarquer une transaction ObjectGrid à l'aide de l'objet de session :
    1. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection.
    2. Extrayez la session com.ibm.websphere.objectgrid.Session.
    3. Utilisez la méthode Session.begin() pour démarrer la transaction.
    4. Utilisez la session pour interagir avec la grille de données.
    5. Utilisez les méthodes Session.commit() ou rollback() pour mettre fin à la transaction.
    6. Fermez la connexion.
- Utilisez une transaction javax.resource.cci.LocalTransction pour démarquer une transaction locale. Suivez les étapes ci-dessous pour démarquer une transaction ObjectGrid à l'aide de l'interface javax.resource.cci.LocalTransaction :
    1. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection.
    2. Extrayez la transaction javax.resource.cci.LocalTransaction à l'aide de la méthode XSConnection.getLocalTransaction().
    3. Utilisez la méthode LocalTransaction.begin() pour démarrer la transaction.

4. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session pour interagir avec la grille de données.
5. Utilisez les méthodes LocalTransaction.commit() ou rollback() pour mettre fin à la transaction.
6. Fermez la connexion.

- Inscrivez la connexion dans une transaction globale. Cette procédure s'applique également aux transactions gérées par conteneur :
  1. Commencez la transaction globale à l'aide de l'interface javax.transaction.UserTransaction ou d'une transaction gérée par conteneur.
  2. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection.
  3. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session.
  4. Fermez la connexion.
  5. Validez ou annulez la transaction globale.
- 2.5+ Configurez une connexion pour écrire plusieurs partitions dans une transaction. Suivez les étapes ci-dessous pour démarquer une transaction ObjectGrid à l'aide de l'objet de session :
  1. Créez un objet com.ibm.websphere.xs.ra.XSConnectionSpec.
  2. Appelez la méthode XSConnectionSpec et la méthode setMultiPartitionSupportEnabled avec l'argument `true`.
  3. Extrayez la connexioncom.ibm.websphere.xs.ra.XSConnection pour envoyer XSConnectionSpec à la méthode ConnectionFactory.getConnection.
  4. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session.

Exemple

Reportez-vous à l'exemple de code suivant, qui illustre les étapes précédentes de démarcation des transactions eXtreme Scale.

```
    // (C) Copyright IBM Corp. 2001, 2012.
    // All Rights Reserved. Eléments sous licence - Propriété d'IBM.
    package com.ibm.ws.xs.ra.test.ee;

    import javax.naming.InitialContext;
    import javax.resource.cci.Connection;
    import javax.resource.cci.ConnectionFactory;
    import javax.resource.cci.LocalTransaction;
    import javax.transaction.Status;
    import javax.transaction.UserTransaction;

    import junit.framework.TestCase;

    import com.ibm.websphere.objectgrid.ObjectMap;
    import com.ibm.websphere.objectgrid.Session;
    import com.ibm.websphere.xs.ra.XSConnection;

    /**
    * Cet exemple doit être exécuté dans un contexte J2EE sur votre serveur
    * d'applications. Par exemple, en utilisant le servlet d'infrastructure préfabriquée
JUnitEE.
    *
    * Le code présent dans ces méthodes de texte réside généralement sur votre propre
servlet,
    * sur votre EJB ou sur un autre composant Web.
    *
    * L'exemple dépend d'une fabrique de connexions WebSphere eXtreme Scale configurée
    * enregistrée sous le nom JNDI de "eis/embedded/wxscf" qui définit une
    * connexion à une grille contenant une mappe portant le nom "Map1".
    *
    * Cet exemple effectue une recherche directe du nom JNDI et ne nécessite pas une
    * injection de ressource.
    */
    public class DocSampleTests extends TestCase {
        public final static String CF_JNDI_NAME = "eis/embedded/wxscf";
        public final static String MAP_NAME = "Map1";

        Long                key = null;
```

```
        Long                value = null;
        InitialContext      ctx = null;
        ConnectionFactory   cf = null;

        public DocSampleTests() {
        }
        public DocSampleTests(String name) {
            super(name);
        }
        protected void setUp() throws Exception {
            ctx = new InitialContext();
            cf = (ConnectionFactory)ctx.lookup(CF_JNDI_NAME);
            key = System.nanoTime();
            value = System.nanoTime();
        }
        /**
         * Cette exemple s'exécute lorsqu'il n'est pas dans le contexte d'une transaction
globale
         * et utilise la validation automatique (autocommit).
         */
        public void testLocalAutocommit() throws Exception {
            Connection conn = cf.getConnection();
            try {
                Session    session = ((XSConnection)conn).getSession();
                ObjectMap  map = session.getMap(MAP_NAME);
                map.insert(key, value); // Or various data access operations
            }
            finally {
                conn.close();
            }
        }

        /**
         * Cette exemple s'exécute lorsqu'il n'est pas dans le contexte d'une transaction
globale
         * et démarque la transaction à l'aide de session.begin()/session.commit()
         */
        public void testLocalSessionTransaction() throws Exception {
            Session    session = null;
            Connection conn = cf.getConnection();
            try {
                session = ((XSConnection)conn).getSession();
                session.begin();
                ObjectMap  map = session.getMap(MAP_NAME);
                map.insert(key, value); // Or various data access operations
                session.commit();
            }
            finally {
                if (session != null && session.isTransactionActive()) {
                    try { session.rollback(); }
                    catch (Exception e) { e.printStackTrace(); }
                }
                conn.close();
            }
        }

        /**
         * Cet exemple utilise l'interface LocalTransaction pour démarquer les
         * transactions.
         */
        public void testLocalTranTransaction() throws Exception {
            LocalTransaction tx = null;
            Connection conn = cf.getConnection();
            try {
                tx = conn.getLocalTransaction();
                tx.begin();
                Session    session = ((XSConnection)conn).getSession();
                ObjectMap  map = session.getMap(MAP_NAME);
```

```
            map.insert(key, value); // Or various data access operations
            tx.commit(); tx = null;
        }
        finally {
            if (tx != null) {
                try { tx.rollback(); }
                catch (Exception e) { e.printStackTrace(); }
            }
            conn.close();
        }
    }

    /**
     * Cet exemple dépend d'une transaction à gestion externe.
     * Cette transaction à gestion externe peut généralement être présente dans
     * un EJB avec des attributs de transaction définis sur REQUIRED ou REQUIRES_NEW.
     * REMARQUE : S'il n'y a AUCUNE transaction globale active, cet exemple s'exécute
en
     *       mode de validation automatique car il ne vérifie pas si une transaction
existe.
     */
    public void testGlobalTransactionContainerManaged() throws Exception {
        Connection      conn = cf.getConnection();
        try {
            Session    session = ((XSConnection)conn).getSession();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
        }
        catch (Throwable t) {
            t.printStackTrace();
            UserTransaction tx =
(UserTransaction)ctx.lookup("java:comp/UserTransaction");
            if (tx.getStatus() != Status.STATUS_NO_TRANSACTION) {
                tx.setRollbackOnly();
            }
        }
        finally {
            conn.close();
        }
    }

    /**
     * Cet exemple montre comment démarrer une nouvelle transaction globale à l'aide
de
     * l'interface UserTransaction. Généralement, le conteneur démarre la transaction
     * globale (par exemple, dans un EJB avec un attribut de transaction défini sur
     * REQUIRES_NEW), mais cet exemple démarre aussi la transaction globale
     * à l'aide de l'API UserTransaction si elle n'est pas actuellement active.
     */
    public void testGlobalTransactionTestManaged() throws Exception {
        boolean         started = false;
        UserTransaction tx = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
        if (tx.getStatus() == Status.STATUS_NO_TRANSACTION) {
            tx.begin();
            started = true;
        }
        // else { called with an externally/container managed transaction }
        Connection conn = null;
        try {
            conn = cf.getConnection(); // Get connection after the global tran starts
            Session    session = ((XSConnection)conn).getSession();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
            if (started) {
                tx.commit(); started = false; tx = null;
            }
        }
        finally {
```

```
                if (started) {
                    try { tx.rollback(); }
                    catch (Exception e) { e.printStackTrace(); }
                }
                if (conn != null) { conn.close(); }
            }
        }
 /**
         /**
         * Cet exemple montre un transaction multipartition.
         */

                                public void
testGlobalTransactionTestManagedMultiPartition() throws Exception {
                                                boolean         started = false;
                                                XSConnectionSpec connSpec = new
XSConnectionSpec();

connSpec.setWriteToMultiplePartitions(true);
                                                UserTransaction tx =
(UserTransaction)ctx.lookup("java:comp/UserTransaction");
                                                if (tx.getStatus() ==
Status.STATUS_NO_TRANSACTION) {
                                                                tx.begin();
                                                started = true;
                                        }
                                // else { called with an externally/container managed
transaction }
                                                Connection conn = null;
                                                                        try {
                                                conn = cf.getConnection(connSpec); // Get
connection after the global tran starts
                                                Session    session =
((XSConnection)conn).getSession();
                                                ObjectMap  map = session.getMap(MAP_NAME);
                                                map.insert(key, value); // Or various data
access operations
                                                if (started) {
                                                                tx.commit(); started =
false; tx = null;
                                                                }
                                                }
                                                finally {
                                                        if (started) {
                                                                        try {
tx.rollback(); }
                                                                        catch (Exception
e) { e.printStackTrace(); }
                                                        }
                                                        if (conn != null) { conn.close();
}
                                                }
                                        }
```

Rubrique parent : 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Utilisation du verrouillage

Les verrous comportent des cycles de vie et leurs différents types sont compatibles entre eux selon plusieurs critères. Les verrous doivent être traités dans un ordre approprié pour éviter les situations d'interblocage.

2.5+ **Configuration et mise en oeuvre du verrouillage dans les applications Java**
Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale.

2.5+ **Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java**
Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Java 2.5+ **Exemple : ordonnancement des verrous avec la méthode flush**
L'appel de la méthode flush sur l'interface ObjectMap avant une validation peut introduire des contraintes supplémentaires concernant l'ordre des verrous. La méthode flush est généralement utilisée pour forcer la transmission des modifications apportées à la mappe au programme d'arrière-plan par l'intermédiaire du plug-in du chargeur.

Java 2.5+ **Exemples Java pour l'isolement de transaction**
L'isolement de transaction définit comment les modifications apportées par une opération deviennent visibles aux autres opérations simultanées. Vous pouvez utiliser les exemples suivants pour définir le niveau d'isolement de transaction dans votre application Java.

Rubrique parent : 2.5+ Programmation de transactions dans des applications Java

Java

Configuration et mise en oeuvre du verrouillage dans les applications Java

Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale.

Avant de commencer

- Déterminez la stratégie de verrouillage à utiliser. Pour plus d'informations, voir Stratégies de verrouillage.
- Vous pouvez également configurer une stratégie de verrouillage en configurant une mappe dynamique. Pour plus d'informations, voir Configuration d'une stratégie de verrouillage.

Pourquoi et quand exécuter cette tâche

Pour éviter une exception java.lang.IllegalStateException, vous devez appeler la méthode setLockStrategy avant d'appeler les méthodes initialize ou getSession sur l'instance ObjectGrid.

Procédure

1. Configurez une stratégie de verrouillage dans votre application Java™.
   - Configurez une stratégie de verrouillage optimiste. Utilisez pour cela la méthode setLockStrategy :

     ```
     import com.ibm.websphere.objectgrid.BackingMap;
     import com.ibm.websphere.objectgrid.LockStrategy;
     import com.ibm.websphere.objectgrid.ObjectGrid;
     import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
     ...
     ObjectGrid og =
             ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test"
     );
     BackingMap bm = og.defineMap("optimisticMap");
     bm.setLockStrategy( LockStrategy.OPTIMISTIC );
     ```

   - Configurez une stratégie de verrouillage pessimiste. Utilisez pour cela la méthode setLockStrategy :

     ```
     import com.ibm.websphere.objectgrid.BackingMap;
     import com.ibm.websphere.objectgrid.LockStrategy;
     import com.ibm.websphere.objectgrid.ObjectGrid;
     import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
     ...
     ObjectGrid og =
             ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test"
     );
     BackingMap bm = og.defineMap("pessimisticMap");
     bm.setLockStrategy( LockStrategy.PESSIMISTIC);
     ```

   - Configurez une stratégie sans verrouillage. Utilisez pour cela la méthode setLockStrategy :

     Remarque : Les mappes de sauvegarde configurées pour utiliser une stratégie sans verrouillage ne peuvent pas participer à une transaction multipartition.

     ```
     import com.ibm.websphere.objectgrid.BackingMap;
     import com.ibm.websphere.objectgrid.LockStrategy;
     import com.ibm.websphere.objectgrid.ObjectGrid;
     import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
     ...
     ObjectGrid og =
             ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test"
     );
     BackingMap bm = og.defineMap("noLockingMap");
     bm.setLockStrategy( LockStrategy.NONE);
     ```

2. Configurez un délai de verrouillage. Utilisez la méthode setLockTimeout sur l'instance BackingMap :

```
bm.setLockTimeout( 60 );
```

Le paramètre de la méthode setLockTimeout est un entier primitif Java qui définit le délai en secondes pendant lequel eXtreme Scale attend l'octroi d'un mode de verrouillage. Si le temps d'attente d'une transaction excède le délai d'attente de verrouillage configurée pour la mappe de sauvegarde, une exception com.ibm.websphere.objectgrid.LockTimeoutException est émise.

3. Si vous utilisez une stratégie de verrouillage pessimiste, vous pouvez utiliser la méthode lock pour verrouiller la clé dans la grille de données, ou verrouiller la clé et déterminer si la valeur existe dans la grille de données. Dans les éditions précédentes, vous utilisiez les API get et getForUpdate pour verrouiller des clés dans la grille de données. Toutefois, si vous n'aviez pas besoin des données du client, les performances étaient dégradées lors de l'extraction des objets de valeur potentiellement volumineux vers le client. De plus, la méthode containsKey ne maintenant aucun verrou, vous étiez obligé d'utiliser les méthodes get et getForUpdate pour obtenir les verrous appropriés en cas d'utilisation du verrouillage pessimiste. L'API lock fournit désormais une méthode containsKey lors du maintien du verrou. Etudiez les exemples suivants :
   - Les méthodes suivantes verrouillent la clé dans la mappe et renvoient la valeur true si la clé existe ou la valeur false si la clé n'existe pas :

     ```
     boolean ObjectMap.lock(Object key, LockMode lockMode);
     ```

   - La méthode suivante verrouille une liste de clés dans la mappe et renvoie une liste de valeurs true ou false ; `true` si la clé existe, `false` si elle n'existe pas :

     ```
     List<Boolean> ObjectMap.lockAll(List keys, LockMode lockMode);
     ```

     LockMode est une énumération qui vous permet d'indiquer les clés que vous souhaitez verrouiller à l'aide des valeurs possibles suivantes :
     - Partagé (SHARED), pouvant être mis à niveau (UPGRADEABLE) et exclusif (EXCLUSIVE)

Voici un exemple de définition du paramètre LockMode :

```
session.begin();
map.lock(key, LockMode.UPGRADABLE);
map.upsert();
session.commit()
```

Rubrique parent : 2.5+ Utilisation du verrouillage

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
Configuration d'une stratégie de verrouillage

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Java

Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java™

Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Procédure

1. Intercepter l'exception et afficher le message résultant.

```
try {
...
} catch (ObjectGridException oe) {
System.out.println(oe);
}
```

L'exception suivante s'affiche :

```
com.ibm.websphere.objectgrid.plugins.LockDeadlockException: Message
```

Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

2. Annuler la transaction après une exception :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
boolean activeTran = false;
try
{
    sess.begin();
    activeTran = true;
    Person p = (IPerson)person.get("Lynn");
    // Lynn a fêté son anniversaire, donc nous l'avons vieilli d'1 an.
    p.setAge( p.getAge() + 1 );
    person.put( "Lynn", p );
    sess.commit();
    activeTran = false;
}
finally
{
    if ( activeTran ) sess.rollback();
}
```

Le bloc `finally` dans le fragment de code garantit qu'une transaction est annulée lorsqu'une exception inattendue se produit. Il ne gère pas seulement une exception de type LockDeadlockException mais également les autres exceptions imprévues éventuelles. Il traite le cas où une exception est émise lors d'un appel de la méthode commit. Cet exemple ne constitue pas le seul moyen pour traiter les exceptions inattendues ; il existe des cas où une application souhaite intercepter certaines des exceptions inattendues susceptibles de se produire afin de pouvoir afficher l'une de ses exceptions d'application. Vous pouvez ajouter des blocs catch comme il convient mais l'application doit faire en sorte que le fragment de code ne se ferme pas sans terminer la transaction.

Rubrique parent : 2.5+ Utilisation du verrouillage

Java

Exemple : ordonnancement des verrous avec la méthode flush

L'appel de la méthode flush sur l'interface ObjectMap avant une validation peut introduire des contraintes supplémentaires concernant l'ordre des verrous. La méthode flush est généralement utilisée pour forcer la transmission des modifications apportées à la mappe au programme d'arrière-plan par l'intermédiaire du plug-in du chargeur.

Dans ce cas, le programme d'arrière-plan utilise son propre gestionnaire de verrou pour contrôler les accès simultanés, ce qui fait que l'état d'attente de verrou et l'interblocage peuvent se produire dans le programme d'arrière-plan et non dans le gestionnaire de verrou de WebSphere eXtreme Scale Client. Examinons la transaction suivante :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
boolean activeTran = false;
try
{
    sess.begin();
    activeTran = true;
    Person p = (IPerson)person.get("Lynn");
    p.setAge( p.getAge() + 1 );
    person.put( "Lynn", p );
    person.flush();
    ...
    p = (IPerson)person.get("Tom");
    p.setAge( p.getAge() + 1 );
    sess.commit();
    activeTran = false;
}
finally
{
    if ( activeTran ) sess.rollback();
}
```

Supposons qu'une autre transaction a également mis à jour la personne Tom, a appelé la méthode flush, puis a mis à jour la personne Lynn. Si cette situation se présente, l'entrelacement ci-dessous des deux transactions entraîne une condition d'interblocage de la base de données :

```
Verrou X octroyé à la transaction 1 pour "Lynn" lorsque la méthode flush est exécutée.
Verrou X octroyé à la transaction 2 pour "Tom" lorsque la méthode flush est exécutée.
Verrou X demandé par la transaction 1 pour "Tom" pendant le traitement de la validation.
(La transaction 1 se bloque en attente du verrou acquis par la transaction 2.)
Verrou X demandé par la transaction 2 pour "Lynn" pendant le traitement de la validation.
(La transaction 2 se bloque en attente du verrou acquis par la transaction 1.)
```

Cet exemple montre que l'utilisation de la méthode flush peut provoquer un interblocage dans la base de données plutôt que dans WebSphere eXtreme Scale Client. Cet exemple d'interblocage peut se produire indépendamment de la stratégie de verrouillage utilisée. L'application doit veiller à empêcher ce type d'interblocage lors de l'utilisation de la méthode flush et lorsqu'un chargeur est connecté à la mappe de sauvegarde. L'exemple précédent illustre également pourquoiWebSphere eXtreme Scale Client comporte un mécanisme de délai d'attente de verrou. Une transaction qui attend un verrou de base de données peut patienter alors qu'elle possède un verrou d'entrée de mappe WebSphere eXtreme Scale Client. Les problèmes rencontrés au niveau de la base de données peuvent causer des temps d'attente excessifs pour un mode de verrouillage WebSphere eXtreme Scale Client et entraîner l'émission d'une exception LockTimeoutException.

Rubrique parent : 2.5+ Utilisation du verrouillage

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java

Java

Exemples Java pour l'isolement de transaction

L'isolement de transaction définit comment les modifications apportées par une opération deviennent visibles aux autres opérations simultanées. Vous pouvez utiliser les exemples suivants pour définir le niveau d'isolement de transaction dans votre application Java™.

Lecture reproductible avec verrouillage pessimiste

```
map = session.getMap("Order");
session.setTransactionIsolation(Session.TRANSACTION_REPEATABLE_READ);
session.begin();

// Un verrou S est demandé et maintenu et la valeur est copiée dans
// le cache transactionnel.
Order order = (Order) map.get("100");
// L'entrée est expulsée du cache transactionnel.
map.invalidate("100", false);

// La même valeur est demandée de nouveau. Elle contient déjà le
// verrou, donc la même valeur est extraite et copiée dans le
// cache transactionnel.
Order order2 = (Order) map.get("100");

// Tous les verrous sont annulés après la synchronisation de la transaction
// avec la mappe de cache.
session.commit();
```

Lecture validée avec verrouillage pessimiste

```
map1 = session1.getMap("Order");
session1.setTransactionIsolation(Session.TRANSACTION_READ_COMMITTED);
session1.begin();

// Un verrou S est demandé mais immédiatement annulé et
//la valeur est copiée dans le cache transactionnel.

Order order = (Order) map1.get("100");

// L'entrée est expulsée du cache transactionnel.
map1.invalidate("100", false);

// Une deuxième transaction met à jour la même commande.
// Elle obtient un verrou U, met à jour la valeur et valide.
// L'ObjectGrid obtient correctement le verrou X lors de la
// validation car la première transaction utilise l'isolement lecture
// validée.

Map orderMap2 = session2.getMap("Order");
session2.begin();
order2 = (Order) orderMap2.getForUpdate("100");
order2.quantity=2;
orderMap2.update("100", order2);
session2.commit();

// La même valeur est demandée de nouveau. Cette fois, la valeur doit être
// mise à jour mais elle reflète maintenant la
// nouvelle valeur
Order order1Copy = (Order) map1.getForUpdate("100");
```

Rubrique parent : 2.5+ Utilisation du verrouillage

Concepts associés:
Isolement des transactions

Java

Plug-in d'indexation des données

Selon le type d'index que vous voulez générer, WebSphere eXtreme Scale Client fournit des plug-in intégrés que vous pouvez ajouter à la mappe de sauvegarde pour générer un index.

HashIndex

Le plug-in HashIndex intégré, la classe com.ibm.websphere.objectgrid.plugins.index.HashIndex, est un plug-in MapIndexPlugin que vous pouvez ajouter à la mappe de sauvegarde pour générer des index dynamiques. Cette classe prend en charge à la fois les interfaces MapIndex et MapRangeIndex. La définition et l'implémentation d'index peuvent considérablement améliorer les performances des requêtes.

Accès aux données avec des index (API Index)
Utilisez l'indexation pour améliorer l'accès aux données.

Utilisation des sessions pour accéder aux données de la grille
Les applications peuvent commencer et terminer les transactions par le biais de l'interface Session. L'interface Session permet également d'accéder aux interfaces ObjectMap et JavaMap d'application.

Configuration du plug-in HashIndex
Vous pouvez configurer le plug-in HashIndex intégré (classe com.ibm.websphere.objectgrid.plugins.index.HashIndex) à l'aide d'un programme, ou à l'aide d'un index dynamique.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Java

Accès aux données avec des index (API Index)

Utilisez l'indexation pour améliorer l'accès aux données.

Pourquoi et quand exécuter cette tâche

La classe HashIndex est l'implémentation de plug-in d'indexation intégré qui peut prendre en charge les deux interfaces d'indexation d'application intégrée MapIndex et MapRangeIndex. Vous pouvez également créer vos propres index. Vous pouvez ajouter HashIndex sous la forme d'un index dynamique dans la mappe de sauvegarde, obtenir soit un objet proxy d'index MapIndex ou MapRangeIndex et utiliser cet objet proxy d'index pour rechercher des objets mis en cache.

Si vous souhaitez effectuer une itération dans les clés d'une mappe locale, vous pouvez utiliser l'index par défaut. Cet index ne requiert pas de configuration, mais il doit être utilisé sur le fragment en utilisant un agent ou une instance ObjectGrid extraits de la méthode ShardEvents.shardActivated(ObjectGrid shard).

Remarque : Dans un environnement réparti, si l'objet d'index est obtenu à partir d'un ObjectGrid client, l'index possède un objet de type client et toutes les opérations d'index sont exécutées dans un ObjectGrid de serveur. Si la mappe est partitionnée, les opérations d'indexation sont exécutées dans chaque partition à distance. Les résultats de chaque partition sont fusionnés avant de renvoyer les résultats à l'application. Les performances sont déterminées par le nombre de partitions et la taille des résultats renvoyés par chaque partition. Les performances peuvent être faibles si les deux facteurs sont élevés.

Procédure

Accédez aux clés de mappe et aux valeurs avec des index.

- **Index local :**

  Pour effectuer une itération dans les clés d'une mappe locale, vous pouvez utiliser l'index par défaut. Cet index fonctionne uniquement avec le fragment, en utilisant un agent ou l'instance ObjectGrid extraits de la méthode ShardEvents.shardActivated(ObjectGrid shard). Reportez-vous à l'exemple suivant :

  ```
  MapIndex keyIndex = (MapIndex)
  objMap.getIndex(MapIndexPlugin.SYSTEM_KEY_INDEX_NAME);
  Iterator keyIterator = keyIndex.findAll();
  ```

- **Index dynamiques :**

  Vous pouvez créer et supprimer à tout moment des index dynamiques d'une instance BackingMap. Un index dynamique diffère d'un index statique en ce qu'il peut être créé même après l'initialisation de l'instance ObjectGrid contenante. Contrairement à l'indexation statique, l'indexation dynamique est un processus asynchrone et doit être à l'état prêt (ready) avant d'être utilisée. Cette méthode utilise la même approche pour extraire et utiliser les index dynamiques que pour les index statiques. Vous pouvez supprimer un index dynamique s'il n'est plus utile. L'interface BackingMap comprend deux méthodes pour créer et supprimer des index dynamiques.

  Pour plus d'informations concernant les méthodes createDynamicIndex et removeDynamicIndex, voir API BackingMap.

  ```
  import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
  import com.ibm.websphere.objectgrid.ObjectGridManager;
  import com.ibm.websphere.objectgrid.ObjectGrid;
  import com.ibm.websphere.objectgrid.BackingMap;

         ObjectGridManager ogManager =
  ObjectGridManagerFactory.getObjectGridManager();
                BackingMap bm = og.getMap("person");

         // créez l'index après l'initialisation de l'ObjectGrid sans
  DynamicIndexCallback.
         bm.createDynamicIndex("CODE", true, "employeeCode", null);

         try {
             // Si DynamicIndexCallback n'est pas utilisé, attendre que l'index soit
  prêt.
             // Le délai d'attente dépend de la taille actuelle de la mappe
             Thread.sleep(3000);
         } catch (Throwable t) {
  ```

```
            // ...
        }

        // Lorsque l'index est prêt, les applications peuvent tenter d'obtenir
l'instance d'interface
        // de l'index d'application.
        // Les applications doivent trouver un moyen de vérifier que l'index est
prêt à être utilisé,
        // faute de quoi elles utilisent l'interface DynamicIndexCallback.
        // L'exemple ci-dessous illustre comment attendre que l'index soit prêt
        // Prenez en compte la taille de la mappe dans le délai d'attente total.

        Session session = og.getSession();
        ObjectMap m = session.getMap("person");
        MapRangeIndex codeIndex = null;

        int counter = 0;
        int maxCounter = 10;
        boolean ready = false;
        while (!ready && counter < maxCounter) {
            try {
                counter++;
                codeIndex = (MapRangeIndex) m.getIndex("CODE");
                ready = true;
            } catch (IndexNotReadyException e) {
                // implique que l'index n'est pas prêt, ...
                System.out.println("Index non prêt. Veuillez patienter.");
                try {
                    Thread.sleep(3000);
                } catch (Throwable tt) {
                    // ...
                }
            } catch (Throwable t) {
                // exception inattendue
                t.printStackTrace();
            }
        }

        if (!ready) {
            System.out.println("Index non prêt. Remédier à cette situation.");
        }

        // Utilisez l'index pour exécuter des requêtes
        // Reportez-vous à l'interface MapIndex ou MapRangeIndex pour les opérations
prises en charge.
        // L'attribut d'objet sur lequel repose l'index est EmployeeCode.
        // Supposons que l'attribut EmployeeCode est de type entier : le
        // paramètre transmis aux opérations d'index présente ce type de données.

        Iterator iter = codeIndex.findLessEqual(new Integer(15));

        // supprimez l'index dynamique lorsqu'il n'est plus nécessaire

        bm.removeDynamicIndex("CODE");
        // Fermer la session (facultatif dans les versions 7.1.1 et ultérieures)
pour améliorer les performances
        session.close();
```

Que faire ensuite

Vous pouvez utiliser l'interface DynamicIndexCallback pour obtenir des notifications des événements d'indexation. Pour plus d'informations, voir Interface DynamicIndexCallback.

Interface DynamicIndexCallback
L'interface DynamicIndexCallback est destinée aux applications qui veulent recevoir des notifications pour les événements d'indexation ready, error ou destroy. DynamicIndexCallback est un paramètre facultatif pour la méthode createDynamicIndex de la mappe de sauvegarde. Avec une instance enregistrée DynamicIndexCallback, les applications peuvent exécuter la logique applicative à la

réception de la notification d'un événement d'indexation.

Rubrique parent : Plug-in d'indexation des données

Java

Interface DynamicIndexCallback

L'interface DynamicIndexCallback est destinée aux applications qui veulent recevoir des notifications pour les événements d'indexation ready, error ou destroy. DynamicIndexCallback est un paramètre facultatif pour la méthode createDynamicIndex de la mappe de sauvegarde. Avec une instance enregistrée DynamicIndexCallback, les applications peuvent exécuter la logique applicative à la réception de la notification d'un événement d'indexation.

Evénements d'indexation

Par exemple, l'événement ready signifie que l'index est prêt à être utilisé. Lorsqu'une notification est reçue pour cet événement, une application peut tenter d'extraire et d'utiliser l'instance d'interface de l'index d'application.

Exemple : utilisation de l'interface DynamicIndexCallback

```
BackingMap personBackingMap = ivObjectGrid.getMap("person");
    DynamicIndexCallback callback = new DynamicIndexCallbackImpl();
    personBackingMap.createDynamicIndex("CODE", true, "employeeCode", callback);


    class DynamicIndexCallbackImpl implements DynamicIndexCallback {
        public DynamicIndexCallbackImpl() {
        }

        public void ready(String indexName) {
            System.out.println("DynamicIndexCallbackImpl.ready() -> indexName = " +
indexName);

            // Simulez le comportement d'une application lors de la notification ready.
            // Normalement, l'application doit patienter jusqu'à l'obtention de l'état
ready, puis doit mettre en oeuvre
            // la logique d'utilisation de l'index.
            if("CODE".equals(indexName)) {
                ObjectGridManager ogManager =
ObjectGridManagerFactory.getObjectGridManager();
                ObjectGrid og = ogManager.createObjectGrid( "grid" );
                Session session = og.getSession();
                ObjectMap map = session.getMap("person");
                MapIndex codeIndex = (MapIndex) map.getIndex("CODE");
                Iterator iter = codeIndex.findAll(codeValue);
                // Fermer la session (facultatif dans les versions 7.1.1 et ultérieures)
pour améliorer les performances
                session.close();
                                    }
        }

        public void error(String indexName, Throwable t) {
            System.out.println("DynamicIndexCallbackImpl.error() -> indexName = " +
indexName);
            t.printStackTrace();
        }

        public void destroy(String indexName) {
            System.out.println("DynamicIndexCallbackImpl.destroy() -> indexName = " +
indexName);
        }
    }
```

Rubrique parent : Accès aux données avec des index (API Index)

Java

Utilisation des sessions pour accéder aux données de la grille

Les applications peuvent commencer et terminer les transactions par le biais de l'interface Session. L'interface Session permet également d'accéder aux interfaces ObjectMap et JavaMap d'application.

Chaque instance ObjectMap ou JavaMap est directement associée à un objet Session spécifique. Chaque unité d'exécution souhaitant accéder à eXtreme Scale doit préalablement obtenir une instance Session à partir de l'objet ObjectGrid. Une instance Session ne peut pas être partagée par plusieurs unités d'exécution. WebSphere eXtreme Scale n'utilise aucun stockage local d'unités d'exécution ; cependant, les restrictions de plate-forme risquent de limiter le passage d'une Session d'une unité d'exécution à une autre.

Méthodes

Méthode Get

Une application obtient une instance Session à partir d'un objet ObjectGrid à l'aide de la méthode ObjectGrid.getSession. L'exemple suivant illustre comment obtenir une instance Session :

```
ObjectGrid objectGrid = ...; Session sess = objectGrid.getSession();
```

Une fois l'instance Session obtenue, l'unité d'exécution garde une référence à la session pour son propre usage. L'appel de la méthode getSession plusieurs fois renvoie un nouvel objet Session chaque fois.

Transactions et méthodes de session

Une session peut être utilisée pour commencer, valider ou annuler une transaction. Les opérations sur BackingMaps avec ObjectMaps et JavaMaps sont exécutées plus efficacement dans une transaction Session. Une fois une transaction commencée, toute modification apportée à un ou plusieurs mappes de sauvegarde dans cette transaction est stockée dans un cache de transaction spécial jusqu'à ce que la transaction soit validée. Lorsqu'une transaction est validée, les modifications en attente sont appliquées aux BackingMaps et Loaders et deviennent visibles par tous les clients de cet ObjectGrid.

WebSphere eXtreme Scale permet également de valider automatiquement les transactions. Si une opération ObjectMap est effectuée en dehors du contexte d'une transaction active, une transaction implicite est lancée avant l'opération et la transaction est automatiquement validée avant de rendre le contrôle à l'application.

```
Session session = objectGrid.getSession();
ObjectMap objectMap = session.getMap("someMap");
session.begin();
objectMap.insert("key1", "value1");
objectMap.insert("key2", "value2");
session.commit();
cobjectMap.insert("key3", "value3"); // auto-validation
```

Méthode Session.flush

La méthode Session.flush est utile lorsqu'un chargeur est associé à une BackingMap. La méthode flush appelle le chargeur avec l'ensemble de modifications actuel dans le cache de transaction. Le chargeur applique les changements au dorsal. Les changements ne sont pas validés lorsque la méthode flush est appelée. Si une transaction de session est validée après l'appel de flush, seules les mises à jour postérieures à l'appel de flush sont appliquées au chargeur. Si une transaction de session est annulée après l'appel de la méthode flush, les modifications vidées sont annulées, de même que toutes les autres modifications en attente dans la transaction. Utilisez la méthode flush avec parcimonie car elle limite les opérations de traitement par lots pour un chargeur. L'exemple ci-dessous illustre l'utilisation de la méthode Session.flush :

```
Session session = objectGrid.getSession();
session.begin();
// faites des modifications supplémentaires
...
session.flush(); // envoyez ces modifications vers le programme de chargement, mais
// ne validez pas ces modifications supplémentaires
...
session.commit();
```

Méthode NoWriteThrough

Certaines cartes sont sauvegardées par un chargeur, qui fournit un stockage de persistance aux données dans la mappe. Il est parfois utile de valider les données uniquement dans la mappe eXtreme Scale et de ne pas envoyer les données au chargeur. L'interface de session fournit la méthode beginNoWriteThough dans ce

but. La méthode beginNoWriteThrough commence une transaction comme la méthode begin. Avec la méthode beginNoWriteThrough, lorsque la transaction est validée, les données sont uniquement validées dans la mappe mémoire et elles ne sont pas validées dans le stockage de persistance fourni par le chargeur. Cette méthode est particulièrement utile lors du préchargement des données sur la mappe.

Lors de l'utilisation d'une instance ObjectGrid répartie, la méthode beginNoWriteThrough est utile pour effectuer des modifications dans le cache proche uniquement, sans modifier le cache éloigné du serveur. Si les données sont périmées dans le cache proche, l'utilisation de la méthode beginNoWriteThrough peut permettre d'invalider les entrées sur le cache proche sans les invalider sur le serveur.

L'interface Session fournit également la méthode isWriteThroughEnabled pour déterminer quel type de transaction est actuellement actif.

```
Session session = objectGrid.getSession();
session.beginNoWriteThrough();
// faites des modifications supplémentaires...
session.commit(); // ces modifications ne seront pas envoyées au chargeur
```

Obtention de la méthode d'objet TxID

L'objet TxID est un objet opaque qui identifie la transaction active. Utilisez l'objet TxID pour les applications suivantes :

- Pour effectuer des comparaisons lorsque vous recherchez une transaction spécifique.
- Pour stocker des données partagées entre les objets TransactionCallback et Loader.
- Déterminez si la transaction a été lancée à partir d'une transaction de session qui utilisait un protocole de validation en une ou deux phases. En examinant la sortie TxID.toString(), vous pouvez déterminer si la transaction était destinée à une transaction monopartition ou multipartition. Si la chaîne commence par le mot clé "Local", cela indique qu'il s'agit d'une transaction à une seule partition. Par exemple : `Local-40000139-72B2-C037-E000-1C271366B073` Si la chaîne commence par le mot clé "WXS", cela indique qu'il s'agit d'une transaction multipartition. Par exemple : `WXS-40000139-72B2-BD3A-E000-1C271366B073`

Reportez-vous au plug-in TransactionCallback et aux Loaders pour des informations supplémentaires sur la fonction d'attribut Object.

Méthode de suivi des performances

Si vous utilisez eXtreme Scale dans WebSphere Application Server, il peut être nécessaire de réinitialiser le type de transaction pour le suivi des performances. Vous pouvez définir le type de transaction avec la méthode setTransactionType. Pour plus d'informations sur la méthode setTransactionType, reportez-vous à la section concernant le suivi des performances d'ObjectGrid par le biais de l'infrastructure d'analyse des performances (PMI) de WebSphere Application Server.

Exécution de la méthode LogSequence

WebSphere eXtreme Scale peut propager des ensembles de modifications de mappe en programmes d'écoute ObjectGrid pour répartir les mappes d'une machine virtuelle Java™ à une autre. Pour que le programme d'écoute puisse traiter les LogSequences plus facilement, l'interface Session fournit la méthode processLogSequence. Cette méthode examine chaque LogElement dans l'objet LogSequence et effectue l'opération appropriée, par exemple une insertion, une mise à jour, une invalidation, etc. sur la BackingMap identifiée par LogSequence MapName. Une session ObjectGrid doit être disponible avant l'appel de la méthode processLogSequence. L'application a également la responsabilité d'effectuer les appels de validation ou d'annulation appropriés pour terminer la session. L'auto-validation n'est pas disponible pour cet appel de méthode. Le traitement normal par l'ObjectGridEventListener récepteur au niveau de la JVM distante est de démarrer une session en utilisant la méthode beginNoWriteThrough, qui empêche la propagation sans fin des modifications , puis d'appeler la méthode processLogSequence, et enfin de valider et d'annuler la transaction.

```
// Utilisez l'objet Session transmis au cours de
//ObjectGridEventListener.initialization...
session.beginNoWriteThrough();
// traitez la LogSequence reçue
try {
        session.processLogSequence(receivedLogSequence);
} catch (Exception e) {
        session.rollback(); throw e;
}
// validez les modifications
session.commit();
```

Méthode markRollbackOnly

Cette méthode est utilisée pour marquer la transaction actuelle en tant que "rollback only" (annulation uniquement). En marquant la transaction "rollback only", vous vous assurez que, même si la méthode de validation est appelée par une application, la transaction est annulée. Cette méthode est généralement utilisée par ObjectGrid lui-même ou par l'application lorsqu'une corruption de données est susceptible de se produire si la transaction est validée. Une fois cette méthode appelée, l'objet Throwable transmis à cette méthode est chaîné à l'exception com.ibm.websphere.objectgrid.TransactionException qui résulte de la méthode de validation si elle est appelée sous une session précédemment marquée comme "rollback only". Tout appel ultérieur de cette méthode pour une transaction déjà marquée en tant que "rollback only" est ignoré. Cela signifie que seul le premier appel transmettant une référence Throwable non nul est utilisé. Une fois la transaction marquée terminée, la marque "rollback only" est supprimée afin que la prochaine transaction lancée par la session soit validée.

Méthode isMarkedRollbackOnly

Renvoie un résultat si Session est marquée "rollback only". Cette méthode renvoie la valeur booléenne True si et uniquement si la méthode markRollbackOnly a été précédemment appelée sur cette Session et si la transaction commencée par cette Session est toujours active.

Méthode setTransactionTimeout

Définissez le délai d'attente de la prochaine transaction démarrée par cette Session sur un nombre spécifique de secondes. Cette méthode n'affecte pas le délai d'attente des transactions précédemment commencées par cette Session. Cela affecte uniquement les transactions lancées après l'appel de la méthode. Si cette méthode n'est jamais appelée, la valeur de délai d'attente passée à la méthode setTxTimeout de la méthode com.ibm.websphere.objectgrid.ObjectGrid est utilisée.

Méthode getTransactionTimeout

Cette méthode renvoie la valeur de délai d'attente de la transaction, exprimée en secondes. La dernière valeur passée en tant que valeur de délai d'attente à la méthode setTransactionTimeout est renvoyée par cette méthode. Si la méthode setTransactionTimeout n'est jamais appelée, la valeur de délai d'attente passée à la méthode setTxTimeout de la méthode com.ibm.websphere.objectgrid.ObjectGrid est utilisée.

transactionTimedOut

Cette méthode renvoie une valeur booléenne si le délai d'attente de la transaction actuelle commencée par cette Session a été dépassé.

Méthode isFlushing

La méthode renvoie la valeur booléenne True si et uniquement si toutes les modifications de transaction sont vidées vers le plug-in Loader comme conséquence de la méthode de vidage de l'interface Session appelée. Cette méthode peut être utile à un plug-in Loader lorsqu'il doit savoir pourquoi la méthode batchUpdate a été appelée.

Méthode isCommitting

Cette méthode renvoie la valeur booléenne True si et uniquement si toutes les modifications de transaction sont validées comme conséquence de la méthode de validation de l'interface de Session appelée. Cette méthode peut être utile à un plug-in Loader lorsqu'il doit savoir pourquoi la méthode batchUpdate a été appelée.

Méthode setRequestRetryTimeout

Cette méthode définit, en millisecondes, la valeur de délai d'attente avant la prochaine tentative de requête pour la session. Si le client définit une valeur de délai d'attente avant la prochaine tentative de requête, le paramètre de la session remplace la valeur du client.

Méthode getRequestRetryTimeout

Cette méthode récupère le paramètre de délai d'attente avant la prochaine tentative de requête sur la session. La valeur `-1` indique que le délai d'attente n'est pas défini. La valeur `0` indique qu'il est en mode d'interruption immédiate. Une valeur supérieure à `0` indique le paramètre de délai d'attente, en millisecondes.

Rubrique parent : Plug-in d'indexation des données

Java

Configuration du plug-in HashIndex

Vous pouvez configurer le plug-in HashIndex intégré (classe com.ibm.websphere.objectgrid.plugins.index.HashIndex) à l'aide d'un programme, ou à l'aide d'un index dynamique.

Pourquoi et quand exécuter cette tâche

La configuration d'un index composite s'effectue de la même manière que celle d'un index standard avec XML, à l'exception de la valeur de la propriété **attributeName**. Dans un index composite, la valeur de la propriété **attributeName** est une liste d'attributs séparés par une virgule. Par exemple, la classe de valeur Address a trois attributs : city, state et zipcode. Un index composite peut être défini avec la valeur de la propriété **attributeName** `"city,state,zipcode"` pour indiquer que la ville, l'état et le code postal sont inclus dans l'index composite.

Notez également qu'un index HashIndex composite ne prend pas en charge les recherches de plages et que sa propriété RangeIndex ne peut pas prendre la valeur true.

Procédure

Configuration d'un index composite à l'aide d'un programme. Ne s'applique qu'à un index dynamique.

L'exemple de code suivant crée le même index composite :

```
          HashIndex mapIndex = new HashIndex();
    mapIndex.setName("Address.CityStateZip");
    mapIndex.setAttributeName(("city,state,zipcode"));
    mapIndex.setRangeIndex(true);



BackingMap bm = objectGrid.getMap(“mymap”);
        bm.createDynamicIndex(mapIndex, null);

        try {
            // Si DynamicIndexCallback n'est pas utilisé, attendre que l'index soit prêt.
            // Le délai d'attente dépend de la taille actuelle de la mappe
            Thread.sleep(3000);
        } catch (Throwable t) {
            // ...
        }

        // Lorsque l'index est prêt, les applications peuvent tenter d'obtenir l'instance
d'interface
        // de l'index d'application.
        // Les applications doivent trouver un moyen de vérifier que l'index est prêt à
être utilisé,
        // faute de quoi elles utilisent l'interface DynamicIndexCallback.
        // L'exemple ci-dessous illustre comment attendre que l'index soit prêt
        // Prenez en compte la taille de la mappe dans le délai d'attente total.

        Session session = obectGrid.getSession();
        ObjectMap m = session.getMap("mymap");
        MapRangeIndex codeIndex = null;

        int counter = 0;
        int maxCounter = 10;
        boolean ready = false;
        while (!ready && counter < maxCounter) {
            try {
                counter++;
                codeIndex = (MapRangeIndex) m.getIndex("Address.CityStateZip");
                ready = true;
            } catch (IndexNotReadyException e) {
                // implique que l'index n'est pas prêt, ...
                System.out.println("Index non prêt. Veuillez patienter.");
                try {
                    Thread.sleep(3000);
                } catch (Throwable tt) {
```

```
                    // ...
                }
            } catch (Throwable t) {
                // exception inattendue
                t.printStackTrace();
            }
        }

        if (!ready) {
            System.out.println("Index non prêt. Remédier à cette situation.");
        }
```

Attributs du plug-in HashIndex

Vous pouvez utiliser les attributs suivants pour configurer le plug-in HashIndex

Utilisation d'un index composite

L'index HashIndex composite permet d'améliorer les performances des requêtes et d'éviter des recherches dans les mappes, consommatrices en ressources. Il facilite également les recherches d'objets mis en cache effectuées par l'API HashIndex lorsque les critères de recherche contiennent plusieurs attributs.

2.5+ Utilisation d'un index global

La mise en oeuvre d'un index global peut améliorer les performances de la recherche de données dans un grand environnement partitionné, comportant, par exemple 100 partitions.

Rubrique parent : Plug-in d'indexation des données

Java

Attributs du plug-in HashIndex

Vous pouvez utiliser les attributs suivants pour configurer le plug-in HashIndex

Attributs

Name

Spécifie le nom de l'index. Le nom doit être unique pour chaque mappe. Ce nom est utilisé pour extraire l'objet d'index de l'instance de mappe d'objet pour la mappe de sauvegarde.

AttributeName

Spécifie à l'index les noms des attributs, séparés par des virgules. Pour les index d'accès par zone, les noms d'attributs sont équivalents aux noms des zones. Pour les index d'accès par propriété, les noms d'attributs sont les noms de propriétés compatibles JavaBean. Si un seul nom d'attribut existe, HashIndex est un index d'attribut unique. Si cet attribut est une relation, il est également un index de relation. Si les noms d'attributs recouvrent plusieurs attributs, l'index HashIndex sera un index composite.

FieldAccessAttribute

Utilisé pour les mappes de non-entité. Si la valeur est égale à `true`, l'accès à l'objet s'effectue directement par les zones. S'il n'est pas défini ou a la valeur `false`, la méthode getter de l'attribut est utilisée pour accéder aux données.

GlobalIndexEnabled

Si la valeur est true, l'index global est activé et l'application peut transtyper l'objet d'index extrait en interface MapGlobalIndex.

Lorsque la propriété GlobalIndexEnabled de HashIndex a la valeur " true", la fonction d'index global de HashIndex est activée pour prendre en charge l'interface MapGlobalIndex sur une configuration HashIndex. Elle fournit un moyen efficace de rechercher des données dans un grand environnement partitionné.

POJOKeyIndex

Utilisé pour les mappes de non-entité. Si `true`, l'index introspecte l'objet dans la partie clé de la mappe. Ce paramètre est utile lorsque la clé est une clé composite et que la valeur ne contient pas la clé intégrée. Si l'attribut n'est pas défini ou que sa valeur est `false`, l'index introspecte l'objet dans la partie clé de la mappe.

RangeIndex

Si `true`, l'indexation de plage est activée et l'application peut transtyper l'objet d'index extrait vers l'interface MapRangeIndex. Si la propriété **`RangeIndex`** a la valeur `false`, l'application ne peut transtyper l'objet d'index extrait que vers l'interface MapIndex.

HashIndex à attribut unique ou HashIndex composite ?

Lorsque la propriété **`AttributeName`** de l'index HashIndex contient plusieurs noms d'attribut, l'index HashIndex est un index composite. Dans le cas contraire, si l'index inclut un seul nom d'attribut, il s'agit d'un index à attribut unique. Par exemple, la valeur de la propriété AttributeName d'un HashIndex composite pourrait être `city,state,zipcode`. Dans ce cas, l'index contient trois attributs séparés par des virgules. Si la valeur de la propriété **`AttributeName`** est uniquement `zipcode`, qui n'a qu'un seul attribut, il s'agit d'un index HashIndex à attribut unique.

Les index HashIndex composites constituent un moyen efficace pour consulter des objets mis en cache lorsque les critères de recherche impliquent plusieurs attributs. Toutefois, ils ne prennent pas en charge les index de plage et la propriété RangeIndex doit avoir la valeur `false`.

Pour plus d'informations, voir Utilisation d'un index composite.

HashIndex de relation

Si l'attribut indexé d'un HashIndex à attribut unique est une relation, que ce soit à valeur unique ou à valeurs multiples, l'index HashIndex est un HashIndex de relation. Pour l'index HashIndex de relation, la propriété RangeIndex de l'index HashIndex doit être définie avec la valeur false.

HashIndex de clés

Dans le cas de mappes de non-entité, lorsque la propriété **`POJOKeyIndex`** de HashIndex a la valeur `true`, l'index HashIndex est un index HashIndex de clés et la partie clé de l'entrée est utilisée pour l'indexation. Lorsque la propriété AttributeName du HashIndex n'est pas spécifiée, la totalité de la clé est indexée. Sinon, le HashIndex de clés ne peut être qu'un HashIndex à attribut unique.

HashIndex de plage

Lorsque la propriété RangeIndex de HashIndex a la valeur `true`, l'index HashIndex est un index de plage et il peut prendre en charge l'interface MapRangeIndex. Une implémentation MapRangeIndex prend en charge des fonctions de recherche de données à l'aide des fonctions de plage telles que supérieur à, inférieur à ou les deux, alors qu'un index MapIndex ne prend en charge que les fonctions d'égalité (equals). Pour un index à attribut unique, la propriété **RangeIndex** ne peut avoir la valeur `true` que si l'attribut indexé est de type Comparable. Si l'index à attribut unique est utilisé par la requête, la propriété RangeIndex doit avoir la valeur `true` et l'attribut indexé doit être de type Comparable. Pour l'index HashIndex de relation et l'index HashIndex composite, la propriété RangeIndex doit avoir la valeur `false`.

L'exemple qui précède est un index HashIndex de plage car la propriété RangeIndex a la valeur `true`.

Le tableau qui suit récapitule l'utilisation de l'index de plage.

Tableau 1. Prise en charge de l'index de plage. Indique si les types de HashIndex prennent ou non en charge l'index de plage.

Type de HashIndex	Prise en charge de l'index de plage
HashIndex à attribut unique : la clé ou l'attribut indexé est de type Comparable	Oui
HashIndex à attribut unique : la clé ou l'attribut indexé n'est pas de type Comparable	Non
HashIndex composite	Non
HashIndex de relation	Non

Rubrique parent : Configuration du plug-in HashIndex

Java

Utilisation d'un index composite

L'index HashIndex composite permet d'améliorer les performances des requêtes et d'éviter des recherches dans les mappes, consommatrices en ressources. Il facilite également les recherches d'objets mis en cache effectuées par l'API HashIndex lorsque les critères de recherche contiennent plusieurs attributs.

Amélioration des performances

Un index HashIndex composite constitue un outil rapide et pratique pour rechercher des objets mis en cache lorsque plusieurs attributs sont indiqués dans les critères de recherche. Il prend en charge les recherches de correspondance d'attribut complète, mais pas les recherches de plages.

Remarque : Les index composites ne prennent pas en charge l'opérateur BETWEEN dans le langage de requête ObjectGrid car la prise en charge des plages est obligatoire pour cet opérateur. De même, les opérateurs conditionnels "supérieur à" (>) et "inférieur à" (<) ne fonctionnent pas, pour la même raison.

Configuration d'un index composite

Vous pouvez configurer une indexation composite à l'aide d'un programme en utilisant un index dynamique.

Configuration par programmation

L'exemple suivant crée un index composite.

```
        HashIndex mapIndex = new HashIndex();
    mapIndex.setName("Address.CityStateZip");
    mapIndex.setAttributeName(("city,state,zipcode"));
    mapIndex.setRangeIndex(false);

  BackingMap bm = objectGrid.getMap("mymap");
    bm.createDynamicIndex(mapIndex, null);
```

Notez que la configuration d'un index composite est identique à la configuration d'un index standard avec XML, à l'exception de la valeur de la propriété attributeName. Dans un index composite, cette valeur est une liste d'attributs séparés par des virgules. Par exemple, la classe de valeur Address a trois attributs : city, state et zipcode. Un index composite peut être défini avec la valeur de propriété attributeName `"city,state,zipcode"` indiquant que la ville, l'état et le code postal sont inclus dans l'index composite.

Les index HashIndexes composites ne prennent pas en charge les recherches de plages et la propriété RangeIndex ne peut donc pas avoir la valeur true.

Exécution de recherches avec un index composite

Une fois un index composite configuré, une application peut utiliser la méthode findAll(Object) de l'interface MapIndex pour exécuter des recherches.

2.5+ **Restriction :** MapIndex.EMPTY_VALUE n'est pas pris en charge pour les index globaux composites.

```
Session sess = objectgrid.getSession();
ObjectMap map = sess.getMap("MAP_NAME");
MapIndex codeIndex = (MapIndex) map.getIndex("INDEX_NAME");
Object[] compositeValue = new Object[]{ MapIndex.EMPTY_VALUE,
                        "MN", "55901"};
Iterator iter = mapIndex.findAll(compositeValue);
// Fermer la session (facultatif dans les versions 7.1.1 et ultérieures) pour améliorer
les performances
sess.close();
```

La valeur MapIndex.EMPTY_VALUE est affectée à compositeValue[0] qui indique que l'attribut city est exclu de l'évaluation. Seuls les objets dont l'attribut state est égal à `"MN"` et l'attribut zipcode est égal à `"55901"` figureront dans le résultats.

Migration et interopérabilité

La seule contrainte liée à l'utilisation d'un index composite est qu'une application ne peut pas le configurer dans un environnement réparti contenant des conteneurs hétérogènes. Les anciens et nouveaux serveurs ne peuvent pas être combinés, car les anciens serveurs de conteneur ne reconnaissent pas les configurations d'index composite. L'index composite est semblable à un index d'attribut normal, mais il permet en outre l'indexation de plusieurs attributs. En cas d'utilisation d'un index d'attribut standard, un environnement

composé de plusieurs types de conteneur est viable.

Rubrique parent : Configuration du plug-in HashIndex

Utilisation d'un index global

2.5+ La mise en oeuvre d'un index global peut améliorer les performances de la recherche de données dans un grand environnement partitionné, comportant, par exemple 100 partitions.

Cette fonction fournit également un moyen de rechercher l'emplacement des attributs indexés et peut améliorer les opérations des agents ou des requêtes qui sont associées aux attributs indexés. Reportez-vous à la documentation de l'API MapGlobalIndex pour plus de détails sur les fonctions de l'index global.

Amélioration des performances

Dans un grand environnement partitionné, les objets mis en cache sont répartis dans toutes les partitions. Par conséquent, toute recherche de données utilisant un index, des requêtes ou des agents devra être exécutée sur tous les serveurs pour obtenir des résultats complets. Ce type de recherche est donc lent en raison des appels distants requis pour charger et explorer chaque partition. En outre, toutes les partitions ne contiennent pas des données qui correspondent aux critères de recherche. Un index global améliore les performances de recherche car il n'effectue les recherches que dans les partitions qui contiennent des données qui correspondent. La fonction d'index global peut suivre l'emplacement des attributs indexés et déterminer les partitions applicables pour les attributs parmi toutes les partitions. Généralement, les partitions applicables sont un sous-ensemble de toutes les partitions. Par conséquent, l'exécution des index, des requêtes et des agents sur les partitions applicables est beaucoup plus rapide que leur exécution sur toutes les partitions, même lorsque cela est compensé par l'index global.

Recherche de données

Les applications peuvent rechercher des données en utilisant des clés. Elles peuvent aussi utiliser des index si les données comportent un ou plusieurs attributs pour lesquels des index sont définis. Traditionnellement, les applications peuvent utiliser un proxy d'index client pour obtenir les clés d'entrée de toutes les partitions, ou utiliser un agent pour effectuer une recherche d'index sur toutes les partitions et renvoyer des clés de cache, des valeurs ou les deux. Grâce à la fonction d'index global, les applications peuvent rechercher des clés d'entrée, des valeurs ou les deux via l'API MapGlobalIndex, selon une approche efficace qui n'exécute les opérations que sur les partitions applicables.

Fonctionnement de l'agent

Si une opération d'agent est lié à des attributs indexés (par exemple, invalidation des entrées à l'aide d'attributs indexés), les applications peuvent utiliser l'index global pour rechercher en premier lieu les partitions applicables d'après les attributs. Ensuite, l'application peut envoyer l'agent à ces partitions applicables. Utilisez la méthode MapGlobalIndex.findPartitions() pour rechercher les partitions applicables en utilisant des attributs.

Activation d'un index global

L'index global est une extension du plug-in HashIndex, qui peut être activée sur n'importe quelle configuration HashIndex existante.

Exécution d'une recherche d'index globale

La fonction d'index global est défini dans l'API MapGlobalIndex. Après l'activation de l'index global dans un plug-in HashIndex, l'application peut transtyper un proxy d'index obtenu en type MapGlobalIndex et commencer à l'utiliser.

```
// in client ObjectGrid process
MapGlobalIndex mapGlobalIndexCODE = (MapGlobalIndex)m.getIndex("CODE", false);
Object[] attributes = new Object[] {new Integer(1)};
Collection partitions = mapGlobalIndexCODE.findPartitions(attributes);
Set keys = mapGlobalIndexDependency.findKeys(attributes);
Set values = mapGlobalIndexDependency.findValues(attributes);
Map entries = mapGlobalIndexDependency.findEntries(attributes);
```

Migration et interopérabilité

La seule contrainte liée à l'utilisation de l'index global réside dans le fait qu'une application ne peut pas le configurer dans un environnement réparti contenant des conteneurs hétérogènes. Les anciens et nouveaux serveurs ne peuvent pas être combinés, car les anciens serveurs de conteneur ne reconnaissent pas les configurations avec index global ou avec index global composite.

Pour utiliser un index global ou un index global composite, vous devez arrêter préalablement tous les serveurs de conteneur et les clients d'une application. Ensuite, activez l'index global sur la configuration HashIndex et redémarrez les serveurs de conteneur et les clients.

Rubrique parent : Configuration du plug-in HashIndex

Notification aux clients des mises à jour de mappe en utilisant l'interrogation continue

2.5+ Vous pouvez être notifié dans votre machine JVM (Java™ virtual machine) client de l'insertion ou de la mise à jour d'objets ou d'entrées dans la grille de données.

Avant de commencer

Si vous souhaitez utiliser l'interrogation continue, vous devez activer IBM® eXtremeIO, un mécanisme de transport utilisé pour communiquer entre les serveurs et les clients. Pour plus d'informations sur l'activation d'eXtremeIO, voir Configuration d'IBM eXtremeIO (XIO).

Pourquoi et quand exécuter cette tâche

Lorsque vous développez des applications client qui interagissent avec la grille de données, vous pouvez avoir besoin de requêtes qui extraient automatiquement des résultats en temps réel lorsque de nouvelles entrées qui correspondent aux critères de filtrage sont insérées ou mises à jour. Par exemple, vous développez une application de cotation boursière qui nécessite des mises à jour fréquentes. Ces mises à jour correspondent aux fluctuations des cours. Par conséquent, il est essentiel que l'application soit informée des changements immédiatement afin de fournir des résultats précis et en temps opportun. L'interrogation continue, qui a un faible impact sur la mémoire, peut informer de manière proactive les clients des modifications qui se produisent dans la grille de données.

Procédez comme suit pour programmer les applications client pour qu'elles utilisent l'interrogation continue.

Restriction : Les requêtes qui spécifient un chemin d'attribut de valeur `null` ne sont pas prises en charge si l'objet de valeur n'est pas un type primitif Java, tel qu'une chaîne ou un entier. Lorsque `null` est spécifié,le filtre de requête est utilisé pour interroger l'objet de valeur tout entier.

Procédure

1. Appelez le gestionnaire d'interrogation continue dans l'application client. Par exemple, insérez la ligne de code suivante :

   ```
   ContinuousQueryManager cqMan = ContinuousQueryManagerFactory.getManager(og);
   ```

2. Définissez un filtre ou une chaîne de filtres. Vous pouvez implémenter vos propres filtres ou les filtres de base `AND, OR, LT, GT, EQ`, etc. qui sont fournis. Des identifiants uniques sont affectés aux filtres ou aux chaînes de filtres. Pour plus d'informations sur les filtres pris en charge, voir Accès à la documentation des API Java pour rechercher les API d'interrogation continue.

   L'exemple de code suivant montre une manière d'utiliser le filtre de base equals (EQ). Supposons que la grille de données contienne des objets Customer avec la zone `firstName` et que le filtre renvoie `true` lorsque `firstName` est égal à `Larry`.

   ```
   EQFilter<String, String> equalsFilter = new EQFilter<String, String>("firstName",
   "Larry");
   ```

3. Définissez une requête en utilisant le filtre que vous avez créé au cours de l'étape précédente ; par exemple :

   ```
   ContinuousQueryTopic<String, Customer> topic =
     cqMan.<String, Customer> defineContinuousQuery("myMapName", equalsFilter, true,
   true, true);
   ```

4. Facultatif : Indiquez que le cache de requête doit accéder aux résultats côté client de l'interrogation continue. Si la requête est définie comme requête de clés, seules les clés qui correspondent à la requête se trouvent dans le cache d'interrogation continue. Par exemple :

   ```
   ContinuousQueryCache cache = topic.getCache();
   ```

5. Facultatif : En outre, vous pouvez enregistrer une classe qui implémente l'interface ContinuousQueryListener avec une instance ContinuousQueryTopic pour recevoir la modification de l'interrogation continue. Appelez la méthode addListener pour enregistrer le programme d'écoute. Par exemple :

   ```
   ContinuousQueryListener<String, Customer> listener = new MyCQListener<String,
   Customer>();
   topic.addListener(listener);
   ```

Que faire ensuite

Voir Documentation des API : Package com.ibm.websphere.objectgrid.continuousquery pour plus d'informations sur l'API d'interrogation continue.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Développement d'applications de grille de données avec la passerelle REST

Vous pouvez utiliser la passerelle REST (Representational State Transfer) pour accéder à des grilles de données simples qui sont hébergées par une collectivité. Cette passerelle REST est pratique lorsque vous devez accéder à une grille de données à partir d'environnements non Java.

Avant de commencer

- Vous pouvez utiliser la passerelle REST avec WebSphere DataPower XC10 Appliance version 8.6 ou une version ultérieure.
- Vous devez créer une grille de données simple sur le dispositif. Pour plus d'informations sur la création d'une grille de données simple, voir Création de grilles de données simples.

Pourquoi et quand exécuter cette tâche

Utilisez la passerelle REST pour accéder aux données d'une grille de données simple à partir d'environnements non Java, tels que le dispositif DataPower XI50 ou une application .NET. Vous pouvez aussi utiliser la passerelle REST pour accéder aux données d'une mappe à partir d'une machine virtuelle Java™ qui ne peut pas héberger la fonction ORB IBM® utilisée par l'API ObjectMap basée sur Java.

Transactions

Chaque opération REST sur WebSphere DataPower XC10 Appliance commence et termine une transaction indépendante sur la grille de données. Il n'est pas possible de chaîner ensemble plusieurs opérations dans une même transaction.

Equilibrage de charge

Lorsque vous utilisez la passerelle REST, il incombe aux clients d'équilibrer la charge de leurs requêtes sur la collectivité WebSphere DataPower XC10 Appliance. Vous pouvez utiliser un équilibreur de charge externe ou ajouter de la logique supplémentaire dans le client HTTP que vous utilisez dans le programme client.

Sécurité

Communiquer via la passerelle REST permet toujours de disposer d'une configuration sécurisée, même si la sécurité n'est pas activée sur la grille de données. Configurez les groupes d'utilisateurs qui doivent accéder à la grille de données avec `tous` les droits d'accès à cette grille.

Relation avec le service de données REST WebSphere eXtreme Scale

La passerelle REST est une entité distincte du service de données REST WebSphere eXtreme Scale, qui implémente l'interface de services de données Microsoft ADO.NET.

2.5+ Alias de grille

Lorsque vous devez remplir simultanément plusieurs grilles de données, vous pouvez utiliser la passerelle REST pour créer et gérer un alias de grille. Un alias de grille vous permet de passer d'une grille à une autre et de les remplir simultanément. Par exemple, supposons que vous vouliez effectuer des opérations sur la grille Grille A et que vous vouliez aussi une couche d'adressage indirect pour pouvoir basculer vers une autre grille, Grille B, afin de la remplir. Vous pouvez créer un alias, Grille C, qui pointe vers Grille A. Vous pouvez alors utiliser cet alias pour effectuer des opérations sur Grille A tout en remplissant Grille B. Pendant que vous effectuez les opérations sur Grille A (à l'aide de l'alias Grille C), vous pouvez effectuer une opération REST pour que cet alias pointe vers Grille B. Les alias de grille résident dans la ressource REST `resources/gridalias`, dans laquelle ils peuvent être créés, interrogés et supprimés. Pour plus d'informations concernant la gestion des alias de grille à l'aide de la passerelle REST, voir Passerelle REST : Opérations REST.

Passerelle REST : Format d'URI
En spécifiant un URI dans un format spécifique, vous pouvez accéder à votre grille de données simple et y effectuer des opérations.

Passerelle REST : Format des données
La passerelle REST utilise l'en-tête Content-Type de vos requêtes HTTP pour déterminer le format des données stockées dans la grille de données.

Passerelle REST : Opérations REST
Vous utilisez des opérations HTTP POST, GET et DELETE pour insérer, mettre à jour, obtenir et supprimer des données dans la grille de données. La passerelle REST prend également en charge les requêtes HTTP de gestion d'un alias de grille pointant vers votre grille de données. Les alias de grille sont utiles lorsque vous devez remplir plusieurs grilles simultanément et que vous devez passer de

l'une à l'autre. Il est possible de créer, interroger et supprimer un alias de grille. Ces alias utilisent la ressource REST `/resource/gridalias`.

Exemple de passerelle REST : Insertion et obtention d'entrées de mappe de grille de données
Vous pouvez utiliser les méthodes HTTP POST et GET pour insérer et obtenir des entrées de mappe de grille de données.

Exemple utilisant une passerelle REST : Insertion de données dans une mappe REST et accès à ces données, à partir d'un client Java et à l'aide des API ObjectMap
Lorsque des données sont insérées dans une mappe à l'aide de la passerelle REST, une classe d'encapsuleur de type com.ibm.websphere.xsa.RestValue est utilisée pour encapsuler le type de contenu (content-type) et le corps (body) de requête fournis. Vous pouvez utiliser la même classe RestValue pour insérer des données dans la mappe et obtenir des données de celle-ci à partir d'un client Java et en utilisant les API ObjectMap.

Exemple de passerelle REST : Effacement d'entrées de mappe de grille de données
Vous pouvez utiliser la méthode HTTP DELETE de la passerelle REST pour effacer une mappe dans une grille de données.

Exemple de passerelle REST : Création de mappes dynamiques
Lorsque vous créez une grille de données simple, une mappe par défaut de même nom est créée par défaut. Vous pouvez aussi utiliser des modèles de mappes pour créer d'autres mappes selon les besoins de votre application.

Exemple de passerelle REST : Expiration de la durée de vie (TTL, Time to live)
Vous pouvez définir une valeur de durée de vie pour les mappes last update time (*.LUT) et last access time (*.LAT). La valeur de durée de vie (TTL) par défaut pour les deux types de mappes est d'une heure.

Passerelle REST : Configuration de la sécurité
Pour accéder à une grille de données via la passerelle REST, l'utilisateur doit être authentifié sur WebSphere DataPower XC10 Appliance, que la sécurité de la grille de données ait été activée ou non. Le client d'application doit toujours fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP. Pour accéder à des grilles de données via la passerelle REST, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation.

Passerelle REST : sessions HTTP et cookies
Utilisez des sessions HTTP et des cookies avec la passerelle REST grâce aux en-têtes `Set-Cookie:`.

Rubrique parent : Développement d'applications pour accéder à des grilles de données simples

Tâches associées:
Création de grilles de données simples

Référence associée:
Fichier de propriétés du client
Options de configuration de mappe dynamique

Passerelle REST : Format d'URI

En spécifiant un URI dans un format spécifique, vous pouvez accéder à votre grille de données simple et y effectuer des opérations.

Format d'URI

L'URI REST pour accéder à une grille de données simple sur WebSphere DataPower XC10 Appliance a le format suivant :

```
/resources/datacaches/[nom_grille]/[nom_mappe]/[clé]
```

La racine de contexte par défaut est `resources`.

Si vous créez une grille de données simple nommée `MyDataGrid` sur le dispositif avec le nom d'hôte `myxc10.ibm.com`, l'URL résultante pour accéder au nom de clé `my.data.item` sera :

```
http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyMap/my.data.item
```

Dans l'exemple précédent, la mappe par défaut `MyMap` est utilisée dans la grille `MyDataGrid`. Cette mappe par défaut n'a pas d'éviction de durée de vie. Les entrées placées dans la grille de données y restent tant qu'elles ne sont pas explicitement supprimées. Pour configurer l'éviction de durée de vie, voir Exemple de passerelle REST : Expiration de la durée de vie (TTL, Time to live).

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Passerelle REST : Format des données

La passerelle REST utilise l'en-tête Content-Type de vos requêtes HTTP pour déterminer le format des données stockées dans la grille de données.

Format des données

La passerelle REST utilise l'en-tête Content-type de vos requêtes HTTP pour déterminer le format des données stockées dans la grille de données. Si vous insérez du contenu de type `application/xml`, lorsque votre application effectue une opération GET pour la même clé du cache, le corps de la réponse et le Content-type sont dans un type de format équivalent. Dans cet exemple, le corps de réponse sera au format `application/xml`. Vous pouvez stocker des données de plusieurs types de contenu dans la même grille de données. Voici des exemples de quelques types de contenu valides :

Tableau 1. Types de contenu pour l'en-tête content-type dans des requêtes HTTP

Type de contenu	Utilisez
`application/xml`	XML
`application/json`	Données JavaScript
`application/octet-stream`	Objets sérialisés, données de portée générale

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Passerelle REST : Opérations REST

Vous utilisez des opérations HTTP POST, GET et DELETE pour insérer, mettre à jour, obtenir et supprimer des données dans la grille de données. La passerelle REST prend également en charge les requêtes HTTP de gestion d'un alias de grille pointant vers votre grille de données. Les alias de grille sont utiles lorsque vous devez remplir plusieurs grilles simultanément et que vous devez passer de l'une à l'autre. Il est possible de créer, interroger et supprimer un alias de grille. Ces alias utilisent la ressource REST `/resource/gridalias`.

Opérations REST de remplissage de grilles de données

Tableau 1. Liste des opérations, accompagnées des méthodes HTTP équivalentes et des définitions des codes de réponse

Opération	Méthode HTTP	Code de réponse
Insérer ou mettre à jour	POST	• `200 CREATED` : Les données ont été insérées ou mises à jour correctement dans la grille de données. • `400 BAD REQUEST` : L'opération d'insertion ou de mise à jour des données ne s'est pas terminée correctement.
Obtenir	GET	• `200 OK` : Le corps et le type de contenu de la réponse sont extraits d'une opération d'insertion ou de mise à jour antérieure. • `404 NOT FOUND` : La clé spécifiée n'est pas présente dans la grille de données. • `400 BAD REQUEST` : Le dispositif n'a pas pu traiter la demande.
Supprimer	DELETE	• `200 NO CONTENT` : L'entrée a été supprimée de la grille de données. • `400 BAD REQUEST` : Le dispositif n'a pas pu traiter la demande.

2.5+

Opérations REST de gestion d'un alias de grille de données

Tableau 2. Liste des opérations, accompagnées des méthodes HTTP équivalentes et des définitions des codes de réponse

Opération	Méthode HTTP	Code de réponse
Ajouter ou mettre à jour un alias	POST `/resources/gridalias/<nom_alias>?src=<nom_grille_source>`	• `204 SUCCESS` : L'alias de grille a été créé. • `401 SOURCE GRID DOES NOT EXIST` : L'alias n'a pas pu être créé car la grille de données vers laquelle il pointe n'existe pas. • `409 DATA GRID EXISTS` : L'alias n'a pas pu être créé car une grille de données portant ce nom existe déjà.
Obtenir la grille en cours pour un alias	GET `/resources/gridalias/<nom_alias>`	• `200 OK`
Obtenir la liste de tous les alias	GET `/resources/gridalias/`	• `200 OK`
Supprimer un alias	DELETE `/resources/gridalias/<nom_alias>`	• `204 SUCCESS` : L'alias a été supprimé. • `404 BAD REQUEST` : L'alias n'existe pas.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Insertion et obtention d'entrées de mappe de grille de données

Vous pouvez utiliser les méthodes HTTP POST et GET pour insérer et obtenir des entrées de mappe de grille de données.

Exemple : Opération d'insertion

A l'aide de l'URI et du format de données définis, vous pouvez insérer des informations dans la grille de données. L'exemple suivant insère une clé "bob" dans la grille MyGrid et la mappe MyGrid :

```
POST /resources/datacaches/MyGrid/MyGrid/bob
Content-type: application/xml
<mesdonnées>des données</mesdonnées>
```

Exemple : Opération d'obtention

Pour récupérer cette clé qui a été insérée dans l'exemple précédent, vous pouvez utiliser l'URI suivant :

```
GET /resources/datacaches/MyGrid/MyGrid/bob
```

Vous devez exécuter les opérations GET sur une clé individuelle. Vous ne pouvez pas récupérer toutes les entrées de la mappe.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple utilisant une passerelle REST : Insertion de données dans une mappe REST et accès à ces données, à partir d'un client Java et à l'aide des API ObjectMap

Lorsque des données sont insérées dans une mappe à l'aide de la passerelle REST, une classe d'encapsuleur de type com.ibm.websphere.xsa.RestValue est utilisée pour encapsuler le type de contenu (content-type) et le corps (body) de requête fournis. Vous pouvez utiliser la même classe RestValue pour insérer des données dans la mappe et obtenir des données de celle-ci à partir d'un client Java et en utilisant les API ObjectMap.

Code du client Java permettant d'accéder aux mappes REST

```
RestValue rv = new RestValue();
rv.setContentType(“application/xml”);
String myXml=”<customer>brian</customer>”;
rv.setValue(myXml.getBytes(“UTF8”));
ogSession.begin();
ObjectMap map = ogSession.getMap(“myMap.LUT”);
map.insert(“brian”,rv);
ogSession.commit();
```

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Effacement d'entrées de mappe de grille de données

Vous pouvez utiliser la méthode HTTP DELETE de la passerelle REST pour effacer une mappe dans une grille de données.

Effacement d'une entrée individuelle

Pour supprimer une entrée individuelle, utilisez la méthode DELETE et le nom de clé de l'objet :

```
DELETE http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyDataGrid/my.data.item
```

Effacement d'une mappe entière sur la grille de données

Pour effacer une mappe entière dans la grille de données, utilisez la méthode HTTP DELETE et omettez la partie de l'URI qui concerne la clé. Par exemple, pour effacer la mappe MyDataMap.LUT sur la grille de données MyDataGrid, utilisez l'opération suivante :

```
DELETE http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyDataMap.LUT
```

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Création de mappes dynamiques

Lorsque vous créez une grille de données simple, une mappe par défaut de même nom est créée par défaut. Vous pouvez aussi utiliser des modèles de mappes pour créer d'autres mappes selon les besoins de votre application.

Création de mappe dynamique

La première opération sur une mappe qui correspond au modèle de mappe mais qui n'a pas encore été créée aboutit à la création d'une nouvelle mappe dynamique. Par exemple, pour créer une mappe dynamique en utilisant le modèle `*.LUT` `MyMap.LUT`, vous pouvez utiliser l'URI suivant dans une opération GET, DELETE ou POST :

```
http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyMap.LUT/a.key
```

Pour savoir comment nommer les mappes dynamiques, voir Options de configuration de mappe dynamique.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Expiration de la durée de vie (TTL, Time to live)

Vous pouvez définir une valeur de durée de vie pour les mappes last update time (*.LUT) et last access time (*.LAT). La valeur de durée de vie (TTL) par défaut pour les deux types de mappes est d'une heure.

Exemple

Pour définir une valeur de durée de vie pour les mappes de date/heure de dernière mise à jour (*.LUT) et de date/heure de dernier accès (*.LAT), spécifiez le paramètre de demande de duré de vie avec une valeur exprimée en secondes. Par exemple, pour définir une valeur de durée de vie de 600 secondes sur la clé `a.key`, spécifiez le paramètre de demande `ttl` lorsque la valeur est insérée ou mise à jour dans la grille de données à l'aide de la méthode HTTP POST :

```
http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyMap.LUT/a.key?ttl=600
```

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Passerelle REST : Configuration de la sécurité

Pour accéder à une grille de données via la passerelle REST, l'utilisateur doit être authentifié sur WebSphere DataPower XC10 Appliance, que la sécurité de la grille de données ait été activée ou non. Le client d'application doit toujours fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP. Pour accéder à des grilles de données via la passerelle REST, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation.

Authentification et autorisation

Pour accéder à une mappe de grille de données via une passerelle REST, l'utilisateur ou le groupe d'utilisateurs doit être authentifié et autorisé à accéder à la grille de données spécifiée dans l'URI. Même si la sécurité n'est pas activée sur la grille de données, vous devez configurer le groupe d'utilisateurs que vous utilisez pour communiquer via la passerelle REST pour avoir un accès `intégral` à la grille de données. Pour plus d'informations sur la configuration de l'accès à la grille de données, voir Activation de la sécurité pour les grilles de données. Le client d'application doit fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP.

```
Authorization : Basic <chaîne codée en Base64 de "ID_utilisateur:mot_de_passe">
```

Pour plus d'informations sur le format de l'en-tête d'autorisation de base, voir Wikipédia : Authentification d'accès de base.

Grilles de données sécurisées

Vous pouvez utiliser la passerelle REST dans une configuration de grille de données sécurisée. Pour accéder aux grilles de données sécurisées, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation. L'utilisateur doit être authentifié et autorisé à accéder à la grille de données spécifiée dans l'URI.

Tableau 1. Grilles de données sécurisées

Permission	Get	Post	Supprimer
READ	X		
WRITE	X		
CREATE	X	X	
ALL	X	X	X

Sécurité du transfert

Les clients utilisant la passerelle REST peuvent utiliser le protocole HTTPS si la sécurité du transport est requise.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST
Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Passerelle REST : sessions HTTP et cookies

Utilisez des sessions HTTP et des cookies avec la passerelle REST grâce aux en-têtes `Set-Cookie:`.

La code de la passerelle REST crée une session HTTP lorsqu'elle reçoit une requête d'un client qui ne se trouve pas actuellement en session. Pour éviter la création inutile de sessions et conserver les meilleures performances, le client REST doit conserver les cookies qui sont renvoyés de la passerelle REST avec des en-têtes `Set-Cookie:` et fournir en retour ces mêmes cookies à la passerelle REST avec des en-têtes `Cookie:` lors des requêtes suivantes.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

.NET

Développement d'applications de grille de données avec les API .NET

2.5+ Vous pouvez développer des applications Microsoft .NET qui utilisent la même grille de données que vos applications Java™.

.NET **Configuration de l'environnement de développement .NET**
Pour utiliser WebSphere eXtreme Scale Client for .NET dans Microsoft Visual Studio, vous devez installer l'environnement de développement et configurer le projet pour utiliser l'assemblage WebSphere eXtreme Scale Client for .NET.

.NET **Création de mappes dynamiques avec les API .NET**
Vous pouvez créer des mappes dynamiques avec des API .NET une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

.NET **Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET**
Utilisez des annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes Java et .NET.

.NET **Mappage de clés avec des partitions avec des annotations PartitionKey**
Un alias PartitionKey est utilisé pour identifier les zones ou les attributs sur lesquels un calcul de code de hachage est exécuté pour déterminer la partition dans laquelle les données sont sauvegardées. L'annotation PartitionKey n'est valide que sur les attributs de clé.

.NET **Programmation de transactions dans des applications .NET**
Lorsque vous écrivez une application .NET qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

.NET **Configuration de la sécurité de grille de données pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer .NET et Java pour communiquer via SSL (Secure Sockets Layer) et utiliser la logique d'authentification UserPassword.

.NET **Configuration de TLS pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer Transport Layer Security (TLS) pour WebSphere eXtreme Scale Client for .NET.

.NET **Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET**
Pour envoyer des données d'identification de WebSphere eXtreme Scale Client for .NET au serveur, vous devez implémenter les interfaces ICredentialGenerator et ICredential. Ces interfaces génèrent un objet de données d'identification qui est envoyé à la grille de données et interprété sur le serveur. Sur le serveur, cet objet est interprété par le plug-in correspondant.

.NET **Programmation de données d'identification personnalisées pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez spécifier des données d'identification de l'utilisateur pour une mappe. De cette manière, deux utilisateurs peuvent interagir avec la même grille de données par l'intermédiaire d'une application Web.

Rubrique parent : Développement d'applications pour accéder à des grilles de données simples

.NET

Configuration de l'environnement de développement .NET

Pour utiliser WebSphere eXtreme Scale Client for .NET dans Microsoft Visual Studio, vous devez installer l'environnement de développement et configurer le projet pour utiliser l'assemblage WebSphere eXtreme Scale Client for .NET.

Avant de commencer

- Pour la liste des éditions Microsoft Visual Studio prises en charge, voir Remarques relatives à Microsoft .NET.
- Installez WebSphere eXtreme Scale Client for .NET. Dans l'assistant d'installation, choisissez le chemin **personnalisé** et sélectionnez l'environnement de développement. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client for .NET.

Procédure

1. Ouvrez le projet dans l'environnement Microsoft Visual Studio.
2. Ajoutez une référence à l'assemblage WebSphere eXtreme Scale Client for .NET. L'assemblage se trouve dans le répertoire *net_client_home*\bin. Choisissez le fichier `IBM.WebSphere.Caching.dll`.
3. Ajoutez les lignes suivantes à l'application pour utiliser les API WebSphere eXtreme Scale Client for .NET :

```
using IBM.WebSphere.Caching;
using IBM.WebSphere.Caching.Map;
```

Résultats

Lorsque vous intégrez les assemblages dans l'environnement de développement, IntelliSense est activé pour les API WebSphere eXtreme Scale Client for .NET.

Que faire ensuite

Utilisez WebSphere eXtreme Scale Client pour les API .NET dans l'application client. Pour plus d'information sur l'accès à la documentation des API, voir Accès à la documentation des API WebSphere eXtreme Scale Client for .NET.

.NET **Accès à la documentation des API WebSphere eXtreme Scale Client for .NET**
Vous pouvez accéder à la documentation des API WebSphere eXtreme Scale Client for .NET dans un fichier `.chm` ou en affichant cette documentation dans le centre de documentation.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

.NET

Accès à la documentation des API WebSphere eXtreme Scale Client for .NET

Vous pouvez accéder à la documentation des API WebSphere eXtreme Scale Client for .NET dans un fichier `.chm` ou en affichant cette documentation dans le centre de documentation.

Procédure

Utilisez les options suivantes pour ouvrir la documentation des API WebSphere eXtreme Scale Client for .NET :

- Utilisez la documentation des API .NET Client, installée avec le produit. Pour ouvrir la documentation des API client .NET localement, ouvrez le fichier *net_client_home*`\doc\IBM.WebSphere.Caching.chm`.
- Affichez la documentation des API dans le centre de documentation. Pour plus d'informations, voir la documentation relative au client pour l'API .NET.

Rubrique parent : .NET Configuration de l'environnement de développement .NET

.NET

Création de mappes dynamiques avec les API .NET

2.5+ Vous pouvez créer des mappes dynamiques avec des API .NET une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Avant de commencer

Choisissez les options de configuration que vous souhaitez utiliser dans votre mappe dynamique. Pour plus d'informations, voir Options de configuration de mappe dynamique.

Procédure

Appelez la méthode GetGridMapPessimisticTx.

```
IGridManager gm = GridManagerFactory.GetGridManager( );
ICatalogDomainInfo cdi =
        gm.CatalogDomainManager.CreateCatalogDomainInfo( catalogServerHostsList );
IClientConnectionContext ccc = gm.Connect( cdi, "SimpleClient.properties" );
grid = gm.GetGrid( ccc, "Grid" );
IGridMapPessimisticTx<Object, Object> map =
        grid.GetGridMapPessimisticTx<Object, Object>( "SessionState.LAT.P" );
```

La mappe SessionState.LAT.P une mappe qui utilise l'expulsion en fonction de l'heure de dernier accès, le verrouillage pessimiste et l'invalidation du cache local.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Référence associée:
Options de configuration de mappe dynamique

.NET

Programmation de transactions dans des applications .NET

Lorsque vous écrivez une application .NET qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

.NET **Interaction avec les données dans une transaction pour les applications .NET**
L'API de WebSphere eXtreme Scale Client impose que chaque unité d'exécution possède son propre objet IGridMapPessimisticTx ou IGridMapPessimisticAutoTx. Avec l'objet IGridMapPessimisticTx, la propriété Transaction sert à commencer, valider ou annuler explicitement la transaction. Avec l'objet IGridMapPessimisticAutoTx, les opérations de lancement, de validation et d'annulation de la transaction s'effectuent automatiquement. Utilisez des sessions pour interagir avec les données à l'aide des opérations Add, Put et Replace.

.NET **Configuration et mise en oeuvre du verrouillage dans les applications .NET**
Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste. Vous pouvez également remplacer le délai de verrouillage d'une instance de mappe. Une fois le verrouillage configuré, vous pouvez verrouiller des clés individuelles ou une liste de clés de la mappe.

.NET **Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET**
Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Concepts associés:
Transactions

.NET

Interaction avec les données dans une transaction pour les applications .NET

L'API de WebSphere eXtreme Scale Client impose que chaque unité d'exécution possède son propre objet IGridMapPessimisticTx ou IGridMapPessimisticAutoTx. Avec l'objet IGridMapPessimisticTx, la propriété Transaction sert à commencer, valider ou annuler explicitement la transaction. Avec l'objet IGridMapPessimisticAutoTx, les opérations de lancement, de validation et d'annulation de la transaction s'effectuent automatiquement. Utilisez des sessions pour interagir avec les données à l'aide des opérations Add, Put et Replace.

Pourquoi et quand exécuter cette tâche

Les interfaces IGridMapPessimisticTx et IGridMapPessimisticAutoTx fournissent des opérations telles que Add, Get, Put, Replace et Remove pour manipuler les données. L'interface IGridMapPessimisticTx offre des opérations supplémentaires, telles que Lock et GetAndLock, qui permettent de contrôler les accès simultanés aux données.

Procédure

- Ajout de données.

  La fragment de code suivant montre comment utiliser l'interface IGridMapPessimisticTx pour commencer une nouvelle transaction, créer un élément pour la grille de données, puis valider la transaction :

  ```
  IGridMapPessimisticTx<String,Person> ptmap;
  ptmap = grid.GetGridMapPessimisticTx<String,Person>("PERSON");
  ptmap.Transaction.Begin();
  Person p = new Person();
  p.name = "John Doe";
  ptmap.Add(p.name, p);
  ptmap.Transaction.Commit();
  ```

- Remplacement de données.

  La fragment de code suivant montre comment utiliser l'interface IGridMapPessimisticTx pour commencer une nouvelle transaction, verrouiller un élément dans la grille de données et obtenir sa valeur, remplacer la valeur de cet élément, puis valider la transaction :

  ```
  IGridMapPessimisticTx<String,Person> ptmap;
  ptmap = grid.GetGridMapPessimisticTx<String,Person>("PERSON");
  ptmap.Transaction.Begin();
  Person p = ptmap.GetAndLock("John Doe", LockMode.Upgradable);
  p.age = 30;
  ptmap.Replace(p.name, p);
  ptmap.Transaction.Commit();
  ```

  L'application utilise normalement la méthode GetAndLock plutôt que la simple méthode Get pour verrouiller un enregistrement. La méthode doit être appelée pour fournir la valeur mise à jour à la mappe. Si la méthode Replace n'est pas appelée, la mappe n'est pas modifiée.

Rubrique parent : .NET Programmation de transactions dans des applications .NET

Concepts associés:
Accès aux données et transactions

.NET

Configuration et mise en oeuvre du verrouillage dans les applications .NET

Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste. Vous pouvez également remplacer le délai de verrouillage d'une instance de mappe. Une fois le verrouillage configuré, vous pouvez verrouiller des clés individuelles ou une liste de clés de la mappe.

Avant de commencer

- Déterminez la stratégie de verrouillage à utiliser. Pour plus d'informations, voir Stratégies de verrouillage.
- Configurez une stratégie de verrouillage pessimiste sur une mappe dynamique. Pour plus d'informations, voir Configuration d'une stratégie de verrouillage.

Procédure

1. Configurez une stratégie de verrouillage pessimiste dans la mappe de sauvegarde. WebSphere eXtreme Scale Client for .NET ne prend en charge que la stratégie de verrouillage pessimiste. Pour plus d'informations, voir Configuration d'une stratégie de verrouillage.
2. Remplacez le délai d'attente de verrou d'une seule instance IGridMapPessimisticTx. Utilisez la propriété **`IGridMapPessimisticTx.LockTimeout`** pour remplacer le délai de verrouillage d'une instance IGridMapPessimisticTx spécifique. La nouvelle valeur du délai de verrouillage affecte toutes les transactions démarrées après sa définition. Cette méthode peut être utilisée lorsque des conflits de verrouillage sont possibles ou prévisibles dans les transactions select.
3. Verrouillez des clés individuelles ou une liste de clés de la mappe. Utilisez la méthode Lock pour verrouiller la clé dans la grille de données ou verrouiller la clé et déterminer si la valeur existe dans la grille de données.
   - La méthode suivante verrouille la clé dans la mappe et renvoie la valeur true si la clé existe ou la valeur false si la clé n'existe pas :

     ```
     bool IGridMapPessimisticTx.Lock(Tkey key, LockMode lockMode);
     ```

   - La méthode suivante verrouille une liste de clés dans la mappe et renvoie une liste de valeurs true ou false ; `true` si la clé existe, `false` si elle n'existe pas :

     ```
     IList<bool> IGridMapPessimisticTx.LockAll(IList<TKey> keyList, LockMode
     lockMode);
     ```

     LockMode est une énumération qui vous permet d'indiquer les clés que vous souhaitez verrouiller à l'aide des valeurs possibles suivantes :

     - Partagé (Shared), pouvant être mis à niveau (Upgradeable) et exclusif (Exclusive)

Voici un exemple de définition du paramètre LockMode :

```
ptmap.Transaction.Begin();
ptmap.Lock(key, LockMode.Upgradable);
ptmap.Put(key,value);
ptmap.Transaction.Commit();
```

Rubrique parent : .NET Programmation de transactions dans des applications .NET

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
Configuration d'une stratégie de verrouillage

.NET

Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET

Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Procédure

1. Intercepter l'exception et afficher le message résultant.

   ```
   try {
   ...
   } catch (GridException ge) {
      System.Console.WriteLine(ge.ToString());
   }
   ```

   Lorsqu'une exception LockDeadlockException est émise, elle peut être contenue comme exception interne dans une autre exception. Le fragment de code ci-dessus affiche l'exception de niveau supérieur ainsi que, le cas échéant, la totalité de la chaîne de l'exception interne. Le message d'exception propre à l'exception LockDeadlockException contient des détails concernant le conflit de verrouillage. Pour plus d'informations sur la manière d'interpréter ce message, voir Traitement des problèmes d'interblocage.

   ```
   IBM.WebSphere.Caching.Map.LockDeadlockException: Message
   ```

   Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

2. Annuler la transaction après une exception :

   ```
   IGridMapPessimisticTx<String,Person> ptmap;
   ptmap = grid.GetGridMapPessimisticTx<String,Person>(“PERSON”);
   try {
      ptmap.Transaction.Begin();
      Person p = ptmap.Get("Lynn");
      // Lynn a fêté son anniversaire, donc nous l'avons vieilli d'1 an.    p.Age++;
      ptmap.Put(p.name, p);
      ptmap.Transaction.Commit();
   }
   catch (GridException ge) {
      System.Console.WriteLine(ge.ToString());
   }
   finally {
      if ( ptmap.Transaction.Active )
         ptmap.Transaction.Rollback();
   }
   ```

   Le bloc `finally` du fragment de code garantit qu'une transaction est annulée lorsqu'une exception inattendue se produit. Il ne gère pas seulement une exception de type LockDeadlockException mais également les exceptions imprévues éventuelles. Il traite le cas où une exception est émise lors d'un appel de la méthode commit. Cet exemple ne constitue pas le seul moyen pour traiter les exceptions inattendues ; il existe des cas où une application souhaite intercepter certaines des exceptions inattendues susceptibles de se produire afin de pouvoir afficher l'une de ses exceptions d'application. Vous pouvez ajouter les blocs `catch` dont vous avez besoin, mais l'application doit veiller à ce que le fragment de code ne quitte pas sans terminer la transaction.

Rubrique parent : .NET Programmation de transactions dans des applications .NET

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET
Configuration d'une stratégie de verrouillage

.NET

Configuration de la sécurité de grille de données pour WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer .NET et Java™ pour communiquer via SSL (Secure Sockets Layer) et utiliser la logique d'authentification UserPassword.

Avant de commencer

Vous devez disposer des fichiers `key.jks` et `trust.jks` correspondant à votre environnement.

Procédure

Activez et configurez la sécurité dans les serveurs. Si la sécurité n'est pas encore configurée sur les serveurs, vous pouvez procédez comme suit pour la configurer avec l'exemple d'authentificateur externe.

a. Obtenez les exemples de fichiers de sécurité. Téléchargez les exemples de fichiers dans le fichier `security_extauth.zip` à partir du wiki consacré à WebSphere eXtreme Scale.
   - `xsjaas3.config` : définit la configuration JAAS (Java Authentication and Authorization Service).
   - `sampleKS3.jks` : contient le fichier de clés des valeurs utilisateurs et mot de passe JAAS.
   - `security3.xml` : définit l'authentificateur à utiliser pour la sécurité.

b. Modifiez le fichier `xsjaas3.config` et définissez le chemin d'accès au fichier `sampleKS3.jks`.

c. Si vous voulez générer votre propre fichier de clés privées au lieu d'utiliser le modèle de fichier `sampleKS3.jks`, utilisez l'utilitaire **keytool** pour générer la clé privée.

```
keytool -genkey -alias myalias -keysize 2048 -keystore key.jks -keyalg rsa -dname
"CN=www.mydomain.com" -storepass password -keypass password -validity 3650
```

d. Modifiez `sampleServer.properties` pour activer la sécurité. Le fichier `sampleServer.properties` se trouve dans le répertoire *racine_install_wxs*`\properties`. Supprimez la mise en commentaire et modifiez les valeurs de propriété suivantes :

```
securityEnabled=true
secureTokenManagerType=none
alias=ogsample
contextProvider=IBMJSSE2
protocol=SSL
keyStoreType=JKS
keyStore=../../../../xio.test/etc/test/security/key.jks
keyStorePassword=ogpass
trustStoreType=JKS
trustStore=../../../../xio.test/etc/test/security/trust.jks
trustStorePassword=ogpass
```

Que faire ensuite

Configurez TLS (Transport Layer Security) pour WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de TLS pour WebSphere eXtreme Scale Client for .NET.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

.NET

Configuration de TLS pour WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer Transport Layer Security (TLS) pour WebSphere eXtreme Scale Client for .NET.

Avant de commencer

- Vous devez disposer d'un fichier de clés comportant les mots de passe associés que vous voulez ajouter à la configuration de WebSphere eXtreme Scale Client for .NET.

Procédure

1. Facultatif : A l'aide de l'utilitaire keytool, extrayez le certificat public du fichier `key.jks`.

   ```
   keytool -export -alias myalias -keystore key.jks -file public.cer -storepass
   password
   ```

   Importez cette clé publique vers le magasin de certificats Windows avec l'outil de gestion des certificats, `certmgr.msc`, pour importer la clé dans le dossier des certificats 'Trusted Root Certification Authority' ou 'Trusted People'. (La propriété **keyStore** du fichier `client.properties` pointe vers ce fichier.)

2. Modifiez le fichier `Client.Net.properties` pour y placer les valeurs de propriété suivantes :

   ```
   securityEnabled=true
   credentialAuthentication=supported
   authenticationRetryCount=3
   credentialGeneratorAssembly=IBM.WebSphere.Caching.CredentialGenerator,Version=8.6.0.
   0,
   Culture=neutral,PublicKeyToken=b439a24ee43b0816
   credentialGeneratorProps=manager manager1
   transportType=ssl-required
   publicKeyFile=<name>.cer
   ```

   La valeur de la propriété credentialGeneratorProps, `manager manager1`, est utilisée comme nom d'utilisateur et mot de passe envoyés au serveur dans l'objet Credential.

   La propriété **publicKeyFile** est affectée d'un chemin relatif d'accès à l'environnement d'exécution .NET. Si cette propriété n'est pas définie, le fichier `public.cer` est recherché dans le magasin de certificats Windows. Si elle est définie, le fichier spécifié est utilisé comme fichier de certificat public SSL. Si le fichier spécifié est introuvable, le client .NET tente de trouver un fichier `public.cer` correspondant dans le magasin de certificats.

3. Facultatif : Codez la valeur de la propriété credentialGeneratorProps. Pour ce faire, transférez votre fichier `Client.Net.properties` sur un ordinateur équipé d'une installation Java de WebSphere eXtreme Scale Client . Exécutez l'utilitaire **FilePasswordEncoder** pour coder la propriété credentialGeneratorProps :

   ```
   FilePasswordEncoder.bat Client.Net.Properties credentialGeneratorProps
   ```

   Lorsque vous exécutez cet utilitaire sur un fichier de propriétés, tous les commentaires contenus dans le fichier sont supprimés.

4. Copiez le fichier *net_client_home*`\IBM.WebSphere.Caching.CredentialGenerator.dll` vers le répertoire *net_client_home*`\sample\SimpleClient\bin\<`*nom_configuration*`>`.
5. Générez l'exemple avec le contexte de projet *nom_configuration*. Exécutez l'exemple sur le serveur.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

.NET

Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Pour envoyer des données d'identification de WebSphere eXtreme Scale Client for .NET au serveur, vous devez implémenter les interfaces ICredentialGenerator et ICredential. Ces interfaces génèrent un objet de données d'identification qui est envoyé à la grille de données et interprété sur le serveur. Sur le serveur, cet objet est interprété par le plug-in correspondant.

Pourquoi et quand exécuter cette tâche

Pour exécuter l'authentification, l'application .NET doit implémenter les interfaces suivantes :

- ICredential : Credential représente les données d'identification d'un client, telles qu'une paire ID utilisateur et mot de passe.
- ICredentialGenerator: CredentialGenerator représente une fabrique de données d'identification pour générer les données d'identification.

Lorsqu'une application client .NET se connecte à un serveur qui nécessite l'authentification, le client doit fournir des données d'identification. Les données d'identification d'un client sont représentées par l'interface ICredential. Ces données peuvent être une paire nom-mot de passe, un ticket Kerberos, un certificat client ou des données au format convenu entre le client et le serveur. Cette interface définit explicitement les méthodes equals(Object) et hashCode. Ces deux méthodes sont importantes car les objets Subject authentifiés sont mis en cache par le biais de l'objet Credential en tant que clé sur le serveur. Vous pouvez également générer des données d'identification avec l'interface ICredentialGenerator. Cette interface est utile lorsque les données d'identification peuvent expirer. De nouvelles données d'identification sont générées chaque fois que la propriété Credential est obtenue.

Vous pouvez également utiliser le plug-in d'accréditation CredentialGenerator fourni pour créer des données d'identification basées sur le paramètre **credentialGeneratorProps=** du fichier `Client.Net.Properties`. Les autres paramètres qui définissent le plug-in d'accréditation sont **credentialGeneratorAssembly** et **credentialGeneratorClass**.

Procédure

Implémentez les interfaces ICredentialGenerator et ICredential dans votre application .NET.

A faire : Si vous implémentez ce plug-in d'accréditation du client, vous devez également implémenter un plug-in d'accréditation de serveur capable de recevoir et d'interpréter les données d'identification de WebSphere eXtreme Scale Client for .NET.

Vous pouvez utiliser les exemples suivants pour développer votre application :

- Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET
- Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET

.NET **Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET**
Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredential. Les données d'identification utilisateur/mot de passe stockent un ID utilisateur et un mot de passe.

.NET **Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET**
Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredentialGenerator. L'interface utilise un ID utilisateur et un mot de passe. L'objet UserPasswordCredential contient l'ID utilisateur et le mot de passe obtenus de la propriété Credential accessible en lecture seule.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Référence associée:
.NET Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET
.NET Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET
Fichier de propriétés du client

Information associée:
Interface ICredential
Interface ICredentialGenerator

.NET

Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET

Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredential. Les données d'identification utilisateur/mot de passe stockent un ID utilisateur et un mot de passe.

UserPasswordCredential.cs

```
// Module  :  UserPasswordCrediential.cs

using System;
using IBM.WebSphere.Caching.Security;

namespace com.ibm.websphere.objectgrid.security.plugins.builtins
{
    public class UserPasswordCredential : ICredential
    {
        private String ivUserName;

        private String ivPassword;

        /// <summary>
        ///Creates a UserPasswordCredential with the specified user name and
        /// password.
        ///
        /// ArgumentException if userName or password is null
        /// </summary>
        /// <param name="userName">the user name for this credential</param>
        /// <param name="password">the password for this credential</param>
        public UserPasswordCredential(String userName, String password)
        {
            if (userName == null || password == null) {
                throw new ArgumentException("User name and password cannot be null.");
            }
            this.ivUserName = userName;
            this.ivPassword = password;
        }

        /// <summary>Obtention du nom d'utilisateur de ces données d'identification.
</summary>
        /// <returns>retourne l'argument de nom d'utilisateur envoyé au constructeur
        ///ou la méthode setUserName(String) de cette classe  </returns>
        public String GetUserName() {
            return ivUserName;
        }

        /// <summary>Définir le nom d'utilisateur de ces données d'identification.
        ///ArgumentException if userName is null
        /// </summary>
        /// <param name="userName">userName the user name to set.</param>
        public void SetUserName(String userName) {
            if (userName == null) {
                throw new ArgumentException("User name cannot be null.");
            }
            this.ivUserName = userName;
        }

        /// <summary>Gets the password for this credential.
        /// </summary>
        /// <returns>retourne l'argument de mot de passe envoyé au constructeur ou la
méthode setPassword(String) de cette classe</returns>
        public String GetPassword() {
            return ivPassword;
        }

        /// <summary>Définir le mot de passe de ces données d'identification.
        ///ArgumentException if password is null
```

```
        /// </summary>
        /// <param name="password">the password to set.</param>
        public void SetPassword(String password) {
            if (password == null)
            {
                throw new ArgumentException("Password cannot be null.");
            }
            this.ivPassword = password;
        }

        /// <summary>Vérifie deux objets UserPasswordCredential pour l'égalité.
        ///<p>
        /// Deux objets UserPasswordCredential sont égaux si et seulement si leurs noms
d'utilisateur et mots de passe
        /// sont égaux.
        /// </summary>
        /// <param name="o">Objet dont nous vérifions l'égalité avec cet objet.</param>
        /// <returns> retourne true sur les deux objets UserPasswordCredential sont
équivalents.</returns>
        public bool Equals(ICredential credential)
        {
            if (this == credential) {
                return true;
            }
            if (credential is UserPasswordCredential) {
                UserPasswordCredential other = (UserPasswordCredential)credential;
                return other.ivPassword.Equals(ivPassword) &&
other.ivUserName.Equals(ivUserName);
            }
            return false;
        }

        /// <summary>Retourne le code de hachage de l'objet UserPasswordCredential.
        /// </summary>
        /// <returns>retourne le code de hachage de cet objet</returns>
        public override int GetHashCode() {
            int ret = ivUserName.GetHashCode() + ivPassword.GetHashCode();
            return ret;
        }

        /// <summary>this.Objet comme chaîne
        /// </summary>
        /// <returns>retourne la présentation de chaîne de l'objet UserPasswordCredential
object.</returns>
        public override String ToString() {
            return typeof(UserPasswordCredential).FullName + "[" + ivUserName +
",xxxxxx]";
        }

    }
}
```

Rubrique parent : .NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Information associée:
Interface ICredential
Interface ICredentialGenerator

.NET

Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET

Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredentialGenerator. L'interface utilise un ID utilisateur et un mot de passe. L'objet UserPasswordCredential contient l'ID utilisateur et le mot de passe obtenus de la propriété Credential accessible en lecture seule.

UserPasswordCredentialGenerator.cs

```
// Module  :  UserPasswordCredientialGenerator.cs
//
// Source File Description: Reference Documentation
//
using System;
using System.Security.Authentication;
using IBM.WebSphere.Caching.Security;
using com.ibm.websphere.objectgrid.security.plugins.builtins;

namespace IBM.WebSphere.Caching.Security
{
    public class UserPasswordCredentialGenerator : ICredentialGenerator
    {

        private String ivUser;

        private String ivPwd;

        public ICredential Credential { get { return _getCredential(); } }

        public string Properties { set {_setProperties(value);} }

        public UserPasswordCredentialGenerator()
        {
            ivUser = null;
            ivPwd = null;
        }


        public UserPasswordCredentialGenerator(String user=null, String pwd=null)
        {
            ivUser = user;
            ivPwd = pwd;
        }

        /// <summary>Crée un objet UserPasswordCredential en utilisant le nom
d'utilisateur et le mot de passe de l'objet.
        /// </summary>
        /// <returns>new UserPasswordCredential instance</returns>
        private ICredential _getCredential()
        {
            try
            {
                ICredential MyCredential = new UserPasswordCredential(ivUser, ivPwd) as
ICredential;
                return (ICredential) MyCredential;
            }
            catch (Exception e)
            {
                AuthenticationException CannotGenerateCredentialException = new
AuthenticationException(e.ToString());
                throw CannotGenerateCredentialException;
            }
        }

        /// <summary>Obtient le mot de passe du générateur de données d'identification.
        /// </summary>
        /// <returns> retourne l'argument de mot de passe envoyé au constructeur</returns>
```

```
        public String getPassword()
        {
            return ivPwd;
        }

        /// <summary>Obtient le nom d'utilisateur de ces données d'identification.
        /// </summary>
        /// <returns>retourne l'argument de nom d'utilisateur envoyé au constructeur de
cette classe</returns>
        public String getUserName()
        {
            return ivUser;
        }

        /// <summary>Définit des propriétés supplémentaires, à savoir un nom d'utilisateur
et un mot de passe.
        ///émet une exception ArgumentException si le format n'est pas valide
        /// </summary>
        /// <param name="properties">properties chaîne de propriétés avec un nom
d'utilisateur et un mot de passe séparés par un espace.</param>
        private void _setProperties(string properties)
        {
            String token = properties;
            char[] Seperator = { ' ' };
            String[] StringProperty = properties.Split(Seperator);
            if (StringProperty.Length != 2)
            {
                throw new ArgumentException(
                    "The properties should have a user name and password and separated by
a space.");
            }

            ivUser = StringProperty[0];
            ivPwd = StringProperty[1];
        }

        /// <summary>Vérifie l'égalité de deux objets UserPasswordCredentialGenerator.
        ///<p>
        ///Deux objets UserPasswordCredentialGenerator sont égaux si et seulement si leurs
noms d'utilisateur et mots de passe
        ///sont égaux..
        /// </summary>
        /// <param name="obj">Objet dont nous vérifions l'égalité avec cet objet.</param>
        /// <returns><code>true</code> si les deux objets  UserPasswordCredentialGenerator
sont équivalents</returns>
        public override bool Equals(Object obj)
        {
            if (obj == this)
            {
                return true;
            }

            if (obj != null && obj is UserPasswordCredentialGenerator)
            {
                UserPasswordCredentialGenerator other = (UserPasswordCredentialGenerator)
obj;

                Boolean bothUserNull = false;
                Boolean bothPwdNull = false;

                if (ivUser == null)
                {
                    if (other.ivUser == null)
                    {
                        bothUserNull = true;
                    }
                    else
                    {
```

```
                        return false;
                    }
                }

                if (ivPwd == null)
                {
                    if (other.ivPwd == null)
                    {
                        bothPwdNull = true;
                    }
                    else
                    {
                        return false;
                    }
                }
                return (bothUserNull || ivUser.Equals(other.ivUser)) && (bothPwdNull ||
ivPwd.Equals(other.ivPwd));
            }
            return false;
        }

        /// <summary>Retourne le code de hachage de l'objet
UserPasswordCredentialGenerator.
        /// </summary>
        /// <returns>Retourne le code de hachage de cet objet</returns>
        public override int GetHashCode()
        {
           return ivUser.GetHashCode() + ivPwd.GetHashCode();
        }

    }

}
```

Rubrique parent : .NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Information associée:
Interface ICredential
Interface ICredentialGenerator

.NET

Programmation de données d'identification personnalisées pour WebSphere eXtreme Scale Client for .NET

Vous pouvez spécifier des données d'identification de l'utilisateur pour une mappe. De cette manière, deux utilisateurs peuvent interagir avec la même grille de données par l'intermédiaire d'une application Web.

Procédure

1. Définissez les données d'identification de l'utilisateur dans le fichier `client.properties`.

```
credentialAuthentication=required
authenticationRetryCount=3
credentialGeneratorAssembly=IBM.WebSphere.Caching.CredentialGenerator,
Version=8.6.0.0, Culture=neutral, PublicKeyToken=b439a24ee43b0816
credentialGeneratorClass=IBM.WebSphere.Caching.Security.UserPasswordCredentialGenera
tor
credentialGeneratorProps=manager manager1
```

2. Ajoutez une référence au fichier `IBM.WebSphere.Caching.CredentialGenerator.dll` dans votre projet d'application. Cette DLL de plug-in contient l'implémentation ICredentialGenerator.
3. Utilisez les API ICredentialGenerator dans votre application .NET.

```
//GridManagerFactory.GetGridManager
IGridManager gm = GridManagerFactory.GetGridManager();
//IGridManager.Connect
ICatalogDomainInfo cdi =
gm.CatalogDomainManager.CreateCatalogDomainInfo(hostAndPort);
ctx = gm.Connect(cdi, "client.properties");
//IGridManager.GetGrid
IGrid grid = gm.GetGrid( ctx, "Grid");
ICredentialGenerator credGenManager = new UserPasswordCredentialGenerator("manager",
"manager1");
ICredentialGenerator credGenOperator = new
UserPasswordCredentialGenerator("operator", "operator1");
//IGrid.GetGridMap
IGridMapPessimisticAutoTx<Object, Object> gridMap1  =
grid.GetGridMapPessimisticAutoTx<Object, Object>("Map1", credGenManager);

IGridMapPessimisticAutoTx<Object, Object> gridMap2  =
grid.GetGridMapPessimisticAutoTx<Object, Object>("Map1", credGenOperator);
```

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Surveillance

Vous pouvez contrôler plusieurs aspects de WebSphere DataPower XC10 Appliance, notamment l'état des changements de configuration via la vue **Tâches**, ainsi que les performances de vos grilles de données via la vue **Contrôler** de l'interface utilisateur.

Avant de commencer

Pour afficher des données dans la vue **Tâches** ou la vue **Contrôler** de l'interface utilisateur, votre dispositif doit être configuré et vous devez avoir créé des grilles de données recevant de nouvelles entrées en provenance des applications.

2.5+ **Suivi de l'état de démarrage du dispositif**
Vous pouvez surveiller l'état de démarrage du dispositif. Cela peut vous aider à identifier et résoudre les problèmes éventuels.

Surveillance des grilles de données dans l'interface utilisateur
Vous pouvez utiliser les fonctionnalités de création de graphiques de WebSphere DataPower XC10 Appliance pour afficher les performances globales des grilles de données de votre environnement.

Contrôle des activités à l'aide des tâches
Les tâches permettent de contrôler la progression des modifications administratives (par exemple, les ajouts de dispositif à la collectivité).

Surveillance avec l'utilitaire xscmd
Avec l'utilitaire **`xscmd`**, vous pouvez afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre dispositif.

Surveillance à l'aide de fichiers CSV
Les données de surveillance sont automatiquement consignées dans des fichiers CSV. Ces fichiers CSV peuvent contenir des informations sur les serveurs, la mappe ou la grille de données.

Surveillance avec l'utilitaire xsadmin
Avec l'utilitaire **`xsadmin`**, vous pouvez mettre en forme et afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre WebSphere DataPower XC10 Appliance. Il fournit une méthode pour effectuer l'analyse syntaxique et la détection des données de déploiement actuelles et peut servir de base pour l'écriture d'utilitaires personnalisés.

2.5+ **Surveillance de la santé de l'environnement**
Message Center fournit une vue agrégée des notifications d'événements pour les messages de journal et de l'outil de diagnostic de premier niveau. Vous pouvez afficher ces notifications d'événements avec : Message Center dans la console Web, l'utilitaire **`xscmd`**, les fichiers journaux de santé, l'interface de commande HTTP, ou un programme avec MBeans.

Surveillance avec SNMP (Simple Network Monitoring Protocol)
Avec le support SNMP (Simple Network Monitoring Protocol), vous pouvez contrôler l'état d'un dispositif IBM® WebSphere DataPower XC10 Appliance dans le cadre d'un grand groupe de systèmes dans un centre de données. La surveillance SNMP améliore votre capacité de notifier les problèmes et de les résoudre rapidement.

Configuration de la consignation à distance
Vous pouvez activer la consignation à distance pour enregistrer les entrées de journal sur un serveur distant en dehors du dispositif. La consignation à distance peut être utile lorsque vous devez définir un niveau de journal de débogage détaillé pour isoler un problème ou surveiller un comportement sur une longue période.

Suivi de l'état de démarrage du dispositif

2.5+ Vous pouvez surveiller l'état de démarrage du dispositif. Cela peut vous aider à identifier et résoudre les problèmes éventuels.

Avant de commencer

- Initialisez, démarrez ou redémarrez le dispositif.
- Vous devez avoir accès au nom d'utilisateur et au mot de passe de `xcadmin`.

Pourquoi et quand exécuter cette tâche

Pour suivre l'état de démarrage du dispositif, vous pouvez utiliser le panneau de démarrage de l'interface utilisateur ou la commande **`start-progress`**.

Panneau de l'interface utilisateur consacré au démarrage du dispositif

Ce panneau affiche l'état de démarrage, sous la forme d'un pourcentage d'achèvement et de l'état des différents serveurs et processus. Il est actualisé automatiquement toutes les 15 secondes.

Commande `start-progress`

La commande **`start-progress`** s'exécute à l'aide de l'interface de ligne de commande. Elle renvoie l'état en cours du démarrage. Pour obtenir une version actualisée de cet état, vous devez exécuter à nouveau la commande.

Procédure

- Affichage du panneau d'état de démarrage du dispositif dans l'interface utilisateur.
  1. Utilisez l'URL suivante pour accéder au panneau d'état de démarrage du dispositif :

     ```
     https://<nom_hôte_dispositif>:9443/
     ```

     Le pourcentage d'avancement du démarrage s'affiche.
  2. Si la progression reste bloquée à un certain pourcentage, vous pouvez télécharger les fichiers journaux pour diagnostiquer l'origine du problème. Cliquez sur **Télécharger les fichiers journaux**.
- Affichage de l'état de démarrage du dispositif dans l'interface de ligne de commande.
  1. Connectez-vous à l'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.
  2. Exécutez la commande **`start-progress`**. L'état de la séquence de démarrage du dispositif s'affiche.
  3. Pour actualiser cet état, exécutez de nouveau la commande **`start-progress`**.

Exemple

L'exemple suivant illustre une séquence de démarrage réussie :

```
Le montage du volume 1 a abouti.
Le montage du volume 2 a abouti.
Le service d'administration de grille a démarré.
Le serveur de catalogue a démarré.
Le service de configuration de grille a démarré.
Le serveur de conteneur 01 a démarré.
Le serveur de conteneur 02 a démarré.
Le serveur de conteneur 03 a démarré.
Le serveur de conteneur 04 a démarré.
Le serveur de conteneur 05 a démarré.
Le serveur de conteneur 06 a démarré.
Le serveur de conteneur 07 a démarré.
Le serveur de conteneur 08 a démarré.
La console d'administration a démarré.
```

- Les serveurs de catalogue s'exécutent sur les trois premiers dispositifs de la collectivité. Il se peut que vous ne voyiez pas le message de démarrage du serveur de catalogue si vous utilisez plus de trois dispositifs.
- Le nombre de serveurs de conteneur qui démarrent dépend du type de transport que vous utilisez. Avec IBM eXtremeIO (XIO), 8 serveurs de conteneur démarrent par défaut. Avec ORB (Object Request

Broker), 16 serveurs de conteneur démarrent par défaut. Pour plus d'informations sur les types de transport, voir Configuration d'IBM eXtremeIO (XIO).
- Après l'affichage du message d'état `La console d'administration a démarré.`, vous pouvez vous connecter à l'interface utilisateur.

Que faire ensuite

- Le panneau de démarrage du dispositif indique également l'état Suspendu lorsque vous exécutez les commandes **`request suspend`** et **`request force-suspend`**.

Rubrique parent : Surveillance

Surveillance des grilles de données dans l'interface utilisateur

Vous pouvez utiliser les fonctionnalités de création de graphiques de WebSphere DataPower XC10 Appliance pour afficher les performances globales des grilles de données de votre environnement.

Avant de commencer

Une fois les grilles de données créées et les applications configurées pour les utiliser, les statistiques deviennent disponibles au bout d'un certain temps. Par exemple, avec une grille de données de mémoire cache dynamique, les statistiques deviennent disponibles lorsqu'un WebSphere Application Server exécutant un système de mémoire cache dynamique se connecte à la grille de données de mémoire cache dynamique du dispositif. Si vous utilisez une collectivité, l'initialisation de cette collectivité doit être terminée pour que les statistiques soient disponibles. En général, il suffit d'attendre environ une minute après une modification de configuration importante pour observer le changement au niveau des statistiques.

Pourquoi et quand exécuter cette tâche

Comportement des fonctions graphiques

La série de données Jour, Semaine et Mois persiste à long terme. En revanche, la série de données **Dernière heure** n'est stockée qu'en mémoire. Les données de dernière heure sont stockées aux emplacements suivants :

- En mémoire pour une grille de données
- En mémoire pour le cache de console

Si des données statistiques sont perdues dans un seul de ces emplacements, les données sont toujours disponibles. Si elles sont perdues aux deux emplacements, les données n'apparaissent pas dans le graphique.

Cas de perte des données en mémoire :

- Lors du déplacement de grilles de données, il y a des périodes pendant lesquelles aucune statistique n'est accumulée.
- Lorsque les dispositifs sont redémarrés, les données en mémoire de la grille de données et dans le cache de console sont effacées.
- Lorsque la mémoire du cache de console se remplit, les données utilisées le moins récemment sont effacées de ce cache. Le cache de console contient 2048 entrées de statistiques.
- Lorsque la console est inactive pendant 30 minutes, le processus de console redémarre et la mémoire est effacée.

La collecte des statistiques se poursuit pour la série de données de la dernière heure.

Conseil : Il suffit de déplacer le pointeur de la souris sur n'importe quel point de données du graphique pour afficher des informations plus précises sur ce point.

Procédure

- Pour afficher les performances de toutes vos grilles de données, cliquez sur **Contrôler** > **Généralités sur la grille de données**. Cette page contient les informations suivantes :

  Onglet Capacité utilisée

  Distribution de la capacité actuelle utilisée par les grilles de données

  Ce graphique contient une vue de la capacité totale disponible sur le dispositif ou la collectivité, de la capacité totale utilisée par la copie principale et par les répliques des données, de la capacité limitée restante si une limite de capacité a été définie et des grilles de données qui consomment la partie la plus importante de la capacité. Vous pouvez utiliser les options ci-dessous pour trier les données. Ne sont affichées que les premières 25 grilles de données :

  - Consommatrices de la plus grande capacité
  - Pourcentage le plus élevé de capacité limitée utilisée

  Capacité utilisée sur une période donnée

  Ce graphique représente la capacité utilisée au cours de la période sélectionnée.

Onglet Débit moyen

Les 5 grilles de données les plus actives du point de vue du débit moyen en transactions/seconde

Ce graphique contient la liste des 5 principales grilles de données, classées par débit moyen, mesuré en transactions par seconde.

Débit moyen sur une période donnée

Ce graphique représente le débit moyen, mesuré en transactions par seconde, au cours de la période sélectionnée.

Onglet Délai de transaction moyen sur une période donnée

Les 5 grilles de données les plus lentes du point de vue du délai de transaction moyen sur une période donnée, en millisecondes

Ce graphique contient la liste des cinq grilles de données les plus lentes, classés par délai de transaction moyen sur une période donnée.

Délai de transaction moyen sur une période donnée

Ce graphique représente la durée moyenne des transactions au cours de la période sélectionnée.

- Pour afficher les différentes grilles de données, cliquez sur **Contrôler** > **Généralités sur des grilles de données spécifiques** > ***nom_grille_données***. Cette page affiche un récapitulatif incluant le nombre d'entrées en cache, le délai de transaction moyen, le débit moyen, le taux de réussite en mémoire cache et le pourcentage de capacité limitée au cours des 30 dernières secondes. Vous pouvez également afficher les graphiques suivants :

Capacité utilisée

Ce graphique montre la capacité utilisée du cache par rapport au nombre d'entrées et la limite de capacité configurée du cache. La capacité utilisée porte sur les données primaires et les données répliquées. Vous pouvez modifier l'intervalle affiché : dernière heure, dernier jour, dernière semaine, dernier mois. Le niveau de détail affiché sur le graphique varie en fonction de l'intervalle sélectionné.

Interprétation du nombre d'entrées de la mémoire cache :

- **Pour toutes les grilles de données :** le nombre d'entrées de cache ne représente que les fragments primaires. Pour afficher le nombre d'entrées de cache pour les fragments primaires et les fragments répliques, utilisez la commande **`xscmd -c showMapSizes`**.
- **Pour les grilles de données de mémoire cache dynamique :** 166 entrées de mémoire cache sont créées par défaut pour chaque grille de données de mémoire cache dynamique. Chaque grille de mémoire cache dynamique comporte 83 partitions, et chaque partition ou fragment est initialisé avec deux entrées pour les grilles de mémoire cache dynamique. Par conséquent, le nombre d'entrées de mémoire cache lors de l'initialisation est de 166. Ces entrées de mémoire cache contiennent les statistiques du fournisseur de mémoire cache dynamique et la configuration de la mémoire cache dynamique pour WebSphere Application Server. Par conséquent, 166 entrées de mémoire cache sont affichées dans le panneau de surveillance avant que vous n'ajoutiez des données à la grille de données de mémoire cache dynamique.
- **Pour les grilles de données de session :** Le nombre d'entrées de la grille de données inclut le nombre de sessions, le nombre d'attributs entre toutes les sessions et les entrées de la table d'expulsion. La table d'expulsion est alimentée lorsqu'une session arrivée à expiration n'a pas été invalidée par le conteneur Web. La table d'expulsion n'est alimentée que lors d'une reprise en ligne d'un serveur d'applications.

Utilisation du cache

Ce graphique permet de visualiser le nombre de demandes ayant abouti au cache. Vous pouvez afficher les tentatives de cache, les réussites en mémoire cache et le taux de réussite en mémoire cache dans le graphique.

Débit moyen

Ce graphique montre le nombre moyen de transactions par seconde traitée sur une période et la durée moyenne de chaque transaction.

- Pour afficher plus de détails à propos d'une grille de données spécifique, cliquez sur **Contrôler** > **Rapports détaillés sur la grille de données**. Une arborescence affiche toutes les grilles de données de votre configuration. Vous explorer cette arborescence en aval et accéder à une grille de données spécifique afin d'afficher les mappes appartenant à cette grille de données. Vous pouvez cliquer sur un nom de grille de données ou sur une mappe pour obtenir plus d'informations :

Informations une grille de données

Vous pouvez afficher les capacités utilisées ainsi que la liste des zones auxquelles la grille de données appartient. La capacité utilisée porte sur les données primaires et les données répliquées. Le graphique qui indique la capacité consommée par les 25 principales mappes dans la grille de données s'affiche. Vous pouvez également afficher un pool total qui inclut la capacité par zone. Un graphique indique la capacité de grille de données consommée dans les 25 principales zones.

Informations sur une mappe

Vous pouvez afficher plus d'informations concernant les mappes de chaque grille, notamment les

informations suivantes : nombre total d'entrées de cache primaires de la mappe, débit moyen, durée moyenne de la transaction et capacité totale de la mappe ventilée par rapport aux 25 principales partitions.

Rubrique parent : Surveillance

Information associée:
Leçon 4 du tutoriel du guide de démarrage : Surveillance de l'environnement

Contrôle des activités à l'aide des tâches

Les tâches permettent de contrôler la progression des modifications administratives (par exemple, les ajouts de dispositif à la collectivité).

Pourquoi et quand exécuter cette tâche

Vous pouvez suivre l'état d'une tâche du début à la fin de celle-ci. L'état d'une tâche peut être En file d'attente, En cours d'exécution, Succès ou Echec.

Tableau 1. Icônes d'état des tâches

Icône	Description
En file d'attente ( )	La tâche est entrée dans la file d'attente mais son exécution n'a pas commencé.
En cours d'exécution ( )	La tâche est en cours d'exécution.
Succès ( )	La tâche s'est exécutée sans erreur.
Echec ( )	La tâche ne s'est pas exécutée correctement. Pour plus d'informations concernant l'erreur qui s'est produite, voir les messages dans la tâche.

Chaque tâche est constituée d'un ou plusieurs messages qui fournissent plus d'informations sur son état.

Tableau 2. Icônes de message de tâche

Icône	Description
Information ( )	Le message contient des informations sur une étape de la tâche en cours.
Erreur ( )	La tâche ne s'est pas exécutée correctement. Pour plus d'informations concernant l'erreur qui s'est produite, voir les messages dans la tâche.
2.5+ Avertissement ( )	Le message contient des informations concernant un risque d'échec de la tâche. Mais il se peut que la tâche aboutisse.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Tâches**.
2. Pour afficher une tâche spécifique et obtenir plus d'informations à son sujet, cliquez sur le nom de la tâche. Chaque tâche contient des informations telles que son type, ses dates de début et de fin, et la liste des messages d'état spécifiques permettant de surveiller sa progression. L'état est mis à jour toutes les 10 secondes jusqu'à ce que la tâche se termine, correctement ou non.
3. 2.5+ Facultatif : Suppression des tâches terminées. Une tâche terminée est à l'état Succès ou Echec. Cliquez sur **Supprimer toutes les tâches terminées** ( ). Vous êtes alors invité à confirmer que vous voulez supprimer toutes les tâches terminées. Si vous cliquez sur **OK**, toutes les tâches terminées sont supprimées. Les tâches dont l'état est En file d'attente ou En cours d'exécution ne sont pas supprimées.

Rubrique parent : Surveillance

Tâches associées:
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif

Surveillance avec l'utilitaire `xscmd`

Avec l'utilitaire **`xscmd`**, vous pouvez afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre dispositif.

Avant de commencer

- Voir Administration avec l'utilitaire xscmd pour plus d'informations sur le démarrage de l'utilitaire **`xscmd`**.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'utilitaire **`xscmd`** pour afficher la structure et l'état actuels de la grille de données (par exemple, le contenu de la grille. Dans cet exemple, la structure de grille de données dans cette tâche est une grille de données simple *`ObjectGridA`* avec la mappe *`MapA`* qui appartient au groupe de mappes *`MapSetA`*. Cet exemple montre comment afficher tous les conteneurs dans une grille de données et afficher des mesures filtrées concernant la taille de la mappe *`MapA`*. Pour afficher toutes les options de la commande, exécutez l'utilitaire **`xscmd`** sans arguments ou avec l'option **`-help`**.

Procédure

1. Surveillez l'environnement avec l'utilitaire **`xscmd`**.
   - Pour activer les statistiques pour tous les serveurs, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c setStatsSpec -spec ALL=enabled -g ObjectGridA`
     - Windows `xscmd.bat -c setStatsSpec -spec ALL=enabled -g ObjectGridA`
   - Pour afficher tous les serveurs de conteneur en ligne pour une grille de données, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c showPlacement -g ObjectGridA -ms MapSetA`
     - Windows `xscmd.bat -c showPlacement -g ObjectGridA -ms MapSetA`

     Toutes les informations sur les conteneurs s'affichent.

     Avertissement : Pour obtenir ces informations lorsque le protocole TLS/SSL (Transport Layer Security/Secure Sockets Layer) est activé, vous devez démarrer les serveurs de catalogue et de conteneur avec le port de service JMX défini. Pour définir le port du service JMX, vous pouvez utiliser l'option **`-JMXServicePort`** dans le script **`startOgServer`** ou **`startXsServer`** ou bien appeler la méthode setJMXServicePort dans l'interface ServerProperties.

   - Pour afficher des informations sur les mappes de la grille de données ObjectGridA, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA`
     - Windows `xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA`
   - Pour vous connecter au service de catalogue et afficher des informations sur la mappe MapA pour l'ensemble du domaine de service de catalogue, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA -m MapA -cep CatalogMachine:6645`
     - Windows `xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA -m MapA -cep CatalogMachine:6645`
   - Pour afficher le placement configuré et d'exécution de votre configuration, exécutez la commande suivante :
     - `xscmd -c placementServiceStatus`
     - `xscmd -c placementServiceStatus -g ObjectGridA -ms MapSetA`
     - `xscmd -c placementServiceStatus -ms MapSetA`
     - `xscmd -c placementServiceStatus -g ObjectGridA`

     Vous pouvez définir la portée de la commande pour afficher les informations de placement de l'intégralité de la configuration, une grille de données unique, un groupe de mappes unique ou une combinaison de grille de données et de groupe de mappes
2. Affichez les résumés des états de réplication dans l'environnement.
   - Afficher le résumé des révisions en attente de chaque serveur de conteneur. Vous pouvez exécuter la commande sur un serveur de conteneur spécifique avec l'argument **`-ct`** ou sur tous les serveurs de conteneur si vous n'incluez pas d'argument.
     - UNIX `./xscmd.sh -c showReplicationState -ct container1`
     - Windows `xscmd.bat -c showReplicationState -ct container1`

     Les informations contenues dans la sortie de cette commande inclut la réplication sortante et la réplication entrante. La réplication sortante contient les modifications qui doivent être extraites du fragment primaire sur le serveur de conteneur et placées dans ses fragments réplique sur les autres serveurs de conteneur. La réplication entrante contient les modifications qui doivent être extraites des fragments principaux sur les autres serveur de conteneur et placées dans les

répliques sur le serveur de conteneur. Ces données statistiques peuvent donner une idée de la santé de réplication. Si le nombre de révisions en suspens sur un serveur de conteneur augmente considérablement, des problèmes au niveau du conteneur peuvent exister.

- Affichez le résumé des révisions en attente des fragments entre les domaines de service de catalogue. Vous pouvez exécuter la commande sur un serveur de conteneur spécifique et le domaine de service de catalogue, ou l'intégralité de votre configuration si vous n'incluez pas d'argument.
  - UNIX `./xscmd.sh -c showDomainReplicationState -dom domainA -ct container1`
  - Windows `xscmd.bat -c showDomainReplicationState -dom domainA -ct container1`

  Les informations contenues dans la sortie de cette commande comprend un récapitulatif des révisions en attente de chaque serveur de conteneur pour chaque domaine de service de catalogue lié. La commande renvoie les modifications à répliquer entre chaque fragment primaire et les fragments primaires distants correspondants qui se trouvent dans un autre domaine de service de catalogue.

Rubrique parent : Surveillance

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Surveillance à l'aide de fichiers CSV

Les données de surveillance sont automatiquement consignées dans des fichiers CSV. Ces fichiers CSV peuvent contenir des informations sur les serveurs, la mappe ou la grille de données.

Pourquoi et quand exécuter cette tâche

Les données de surveillance peuvent être consignées par défaut dans des fichiers CSV. Vous pouvez télécharger et analyser des données historiques pour les serveurs exécutés sur le dispositif. La collecte des données commence au démarrage des serveurs. Vous pouvez ensuite télécharger les fichiers CSV à tout moment et utiliser les fichiers comme vous le désirez.

Procédure

1. Téléchargez le fichier CSV. Lors du téléchargement des fichiers journaux à partir du dispositif, les fichiers CSV sont inclus dans le fichier `trace.zip`. Pour télécharger ce fichier, cliquez sur **Dispositif** > **Identification et résolution des incidents** > **Consignation au journal** > **Télécharger les fichiers journaux**. Dans le fichier `trace.zip`, les fichiers CSV se trouvent dans le répertoire *`nom_serveur`*`/logs`. Les fichiers sont intitulés : `jvmstats.log`, `mapstats.log` et `ogstats.log`.
2. Importez le fichier CSV dans le programme que vous utilisez pour traiter les données, par exemple, une feuille de calcul.

Définition des statistiques des fichiers CSV
Les fichiers CSV que vous pouvez télécharger pour un serveur comprennent des statistiques que vous pouvez utiliser pour créer des diagrammes d'historique ou d'autres informations.

Rubrique parent : Surveillance

Tâches associées:
Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance

Référence associée:
Définition des statistiques des fichiers CSV

Définition des statistiques des fichiers CSV

Les fichiers CSV que vous pouvez télécharger pour un serveur comprennent des statistiques que vous pouvez utiliser pour créer des diagrammes d'historique ou d'autres informations.

Journal des statistiques JVM (Java virtual machine)

TimeStamp (colonne 1)

Indique la date et l'heure de l'image instantanée des statistiques, prise pour la machine virtuelle Java (JVM).

ServerName (colonne 2)

Indique le nom du serveur de la machine JVM.

Hostname (colonne 3)

Indique le nom de la machine JVM.

FreeMemory (colonne 4)

Indique le nombre d'octets disponibles pour la machine JVM.

MaxMemory (colonne 5)

Indique le nombre maximal d'octets qui peut être attribué pour la machine JVM.

TotalMemory (colonne 6)

Affiche l'utilisation de la mémoire réelle dans l'environnement d'exécution du serveur.

AvailProcs (colonne 7)

Affiche le nombre de processeurs qui sont disponibles pour ce service de catalogue et ses mappes. Pour une stabilité optimale, vous devez exécuter les serveurs à 60 % du chargement de processeur et les segments de mémoire des machines virtuelles Java à 60 % du chargement des segments de mémoire. Les pics peuvent alors pousser l'utilisation du processeur à 80-90 %, mais ce ne doit pas être le niveau habituel d'exécution de vos serveurs.

Journal des statistiques de mappe

TimeStamp (colonne 1)

Indique la date et l'heure de l'image instantanée des statistiques, prise pour la mappe.

MapName (colonne 2)

Indique le nom de la mappe.

OgName (colonne 3)

Indique le nom de la grille de données à laquelle appartient la mappe.

PartitionId (colonne 4)

Indique l'ID de la partition.

MapSetName (colonne 5)

Indique le groupe de mappes auquel appartient la mappe.

HitRate (colonne 6)

Affiche le taux de réussites pour la mappe sélectionnée. Un taux élevé est souhaitable. Le taux de réussite indique la manière dont la grille de données contribue à éviter d'accéder au stockage de persistance.

Count (colonne 7)

Indique le nombre d'entrées dans la grille de données depuis le démarrage du serveur. Par exemple, la valeur 100 indique que l'entrée est le 100e échantillon collecté depuis le démarrage du serveur.

TotalGetCount (colonne 8)

Affiche le nombre total de fois où la mappe a dû accéder au stockage de persistance pour obtenir des données.

TotalHitCount (colonne 9)

Affiche le nombre total de fois où les données demandées ont été trouvées dans la mappe, dispensant de devoir accéder au stockage de persistance.

StartTime (colonne 10)

Indique l'heure à laquelle l'appel a commencé à partir de la dernière réinitialisation des compteurs. Les réinitialisations se produisent lorsque le serveur démarre ou redémarre.

LastCount (colonne 11)

Indique la durée écoulée depuis le dernier échantillon de données.

LastTotalGetCount (colonne 12)

Indique le nombre total actuel d'opérations d'extraction à partir de la mémoire cache moins le nombre d'opérations d'extraction dans la période précédente.

LastTotalHitCount (colonne 13)

Indique le nombre total actuel d'opérations d'extraction à partir de la mémoire cache moins le nombre d'opérations d'extraction dans la période précédente.

UsedBytes (colonne 14)

Affiche la consommation de la mémoire par cette mappe. Les statistiques d'octets utilisés sont exactes uniquement lorsque vous utilisez des objets simples ou le mode de copie COPY_TO_BYTES.

MinUsedBytes (colonne 15)

Affiche le point bas de la consommation de mémoire par ce service de catalogue et ses mappes. Les statistiques d'octets utilisés sont exactes uniquement lorsque vous utilisez des objets simples ou le mode de copie COPY_TO_BYTES.

MaxUsedBytes (colonne 16)

Affiche le point haut de la consommation de mémoire par ce service de catalogue et ses mappes. Les statistiques d'octets utilisés sont exactes uniquement lorsque vous utilisez des objets simples ou le mode de copie COPY_TO_BYTES.

LastUsedBytes (colonne 17)

Indique la valeur UsedBytes en cours moins la valeur UsedBytes à partir de la période de collecte des statistiques précédentes.

SampleLen (colonne 18)

Indique la durée, en millisecondes, de la période d'échantillonnage des données.

Journal de statistiques ObjectGrid

TimeStamp (colonne 1)

Indique la date et l'heure de l'image instantanée des statistiques, prise pour la grille de données.

OgName (colonne 2)

Indique le nom de la grille de données.

PartitionId (colonne 3)

Indique l'ID de la partition.

Count (colonne 4)

Indique le nombre d'entrées dans la grille de données qui ont été collectées depuis le démarrage du serveur. Par exemple, la valeur 100 indique que l'entrée est le 100e échantillon collecté depuis le démarrage du serveur.

Hostname (colonne 5)

Indique le nom d'hôte.

DomainName (colonne 6)

Indique le domaine du service de catalogue auquel la grille de données appartient.

MaxTime (colonne 7)

Affiche pour ce serveur le temps *maximum* qu'a mis une transaction pour s'exécuter.

MinTime (colonne 8)

Affiche pour ce serveur le temps *minimum* qu'a mis une transaction pour s'exécuter.

MeanTime (colonne 9)

Indique le temps moyen passé sur une transaction.

TotalTime (colonne 10)

Affiche pour ce serveur le temps total passé à des transactions depuis l'initialisation du serveur.

AvgTransTime (colonne 11)

Affiche pour ce serveur la durée moyenne que met une transaction pour s'exécuter.

AvgThroughPut (colonne 12)

Affiche le nombre moyen de transactions par seconde pour ce serveur.

SumOfSquares (colonne 13)

Spécifie la somme des carrés pour le temps de transaction. Cette valeur mesure l'écart par rapport à la moyenne à un moment donné.

SampleLen (colonne 14)

Indique la durée, en millisecondes, de la période d'échantillonnage des données.

LastDataSample (colonne 15)

Indique la durée écoulée depuis le dernier échantillon de données.

LastTotalTime (colonne 16)

Indique le temps total actuel moins le temps total précédent de l'échantillonnage de données.

StartTime (colonne 17)

Indique l'heure à laquelle les statistiques ont commencé à être collectées depuis la dernière réinitialisation des données. Les données sont réinitialisées lorsque le serveur redémarre.

Rubrique parent : Surveillance à l'aide de fichiers CSV

Tâches associées:

Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance
Surveillance à l'aide de fichiers CSV

Surveillance avec l'utilitaire `xsadmin`

Avec l'utilitaire **`xsadmin`**, vous pouvez mettre en forme et afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre WebSphere DataPower XC10 Appliance. Il fournit une méthode pour effectuer l'analyse syntaxique et la détection des données de déploiement actuelles et peut servir de base pour l'écriture d'utilitaires personnalisés.

Avant de commencer

- 2.5+ L'utilitaire **`xsadmin`** est obsolète. Utilisez l'utilitaire **`xscmd`** à la place. L'utilitaire **`xscmd`** est fourni comme utilitaire pris en charge pour la surveillance et l'administration de votre environnement. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Pourquoi et quand exécuter cette tâche

L'utilitaire **`xsadmin`** utilise une implémentation de beans gérés (MBeans). Vous pouvez étendre les capacités de cet utilitaire à l'aide des interfaces du package com.ibm.websphere.objectgrid.management. Vous pouvez consulter le code source de l'application **`xsadmin`** dans le fichier *`rép_base_client_wxs`*`/samples/xsadmin.jar` dans le cas d'une installation autonome, ou bien dans le fichier *`rép_base_client_wxs`*`/optionalLibraries/ObjectGrid/xsadmin.jar` dans le cas d'une installation WebSphere Application Server.

Procédure

1. Téléchargez le fichier de clés certifiées actif pour le dispositif sur le client. Dans l'interface utilisateur de WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres** > **TLS (Transport Layer Security)** > **Télécharger le fichier de clés certifiées actif**. Le fichier de clés certifiées par défaut est le fichier `xsatruststore.jks`. Le mot de passe par défaut est `xc10pass`.
2. Sur la ligne de commande, définissez la variable d'environnement *`JAVA_HOME`*.
   - UNIX `export JAVA_HOME=javaHome`
   - Windows `set JAVA_HOME=javaHome`
3. Accédez au répertoire `bin`.

   ```
   cd rép_base_client_wxs/bin
   ```

4. Exécutez l'utilitaire **`xsadmin`**. Pour vous connecter au dispositif, vous devez inclure les arguments de sécurité pour le fichier de clés certifiées que vous avez téléchargé, le nom d'utilisateur et le mot de passe que vous utilisez pour vous connecter au dispositif, et le nom d'hôte de votre dispositif chaque fois que vous exécutez la commande :

   ```
   xsadmin.sh -trustPath xsatruststore.jks -trustType jks -ssl -trustPass xc10pass
   -username xcadmin -password xcadmin -ch myxc10.mycompany.com
   [additional_xsadmin_parameters]
   ```

   Vous pouvez aussi créer un fichier de configuration pour enregistrer ces paramètres. Voici un exemple de fichier de propriétés avec les paramètres requis :

   ```
   XSADMIN_TRUST_PATH=xsatruststore.jks
   XSADMIN_TRUST_TYPE=JKS
   XSADMIN_TRUST_PASS=xc10pass
   XSADMIN_USERNAME=xcadmin
   XSADMIN_PASSWORD=xcadmin
   ```

   Pour exécuter l'utilitaire **`xsadmin`** avec le fichier de propriétés, utilisez l'argument **`-profile`** pour indiquer l'emplacement du fichier de propriétés.

   ```
   xsadmin.sh -profile myxc10.properties -ssl -ch myxc10.mycompany.com
   [additional_xsadmin_parameters]
   ```

Référence de l'utilitaire xsadmin
Vous pouvez passer des arguments à l'utilitaire **`xsadmin`** selon deux méthodes différentes : avec un argument de ligne de commande ou avec un fichier de propriétés.

Migration de l'outil xsadmin vers l'outil xscmd
Dans les versions précédentes, l'outil **`xsadmin`** était un exemple d'utilitaire de ligne de commande pour surveiller l'état de l'environnement. L'outil **`xscmd`** a été introduit comme outil officiel de ligne de commande d'administration et de surveillance. Si vous utilisiez l'outil **`xsadmin`**, faites migrer les

commandes vers le nouvel outil **`xscmd`**.

Rubrique parent : Surveillance

Référence de l'utilitaire `xsadmin`

Vous pouvez passer des arguments à l'utilitaire **`xsadmin`** selon deux méthodes différentes : avec un argument de ligne de commande ou avec un fichier de propriétés.

Arguments de `xsadmin`

2.5+ **Remarque :** L'utilitaire **`xsadmin`** est obsolète. Utilisez l'utilitaire **`xscmd`** à la place. L'utilitaire **`xscmd`** est fourni comme utilitaire pris en charge pour la surveillance et l'administration de votre environnement. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Vous pouvez définir un fichier de propriétés pour l'utilitaire **`xsadmin`** avec WebSphere eXtreme Scale Client version 7.1 Correctif 1 ou ultérieur. En créant un fichier de propriétés, vous pouvez enregistrer certains des arguments fréquemment utilisés, tels que le nom d'utilisateur. Les propriétés que vous pouvez ajouter à un fichier de propriétés se trouvent dans le tableau ci-dessous. Si vous spécifiez une propriété à la fois dans un fichier de propriétés et dans l'argument de ligne de commande équivalent, la valeur de l'argument de ligne de commande remplace la valeur du fichier de propriétés.

*Tableau 1. Arguments de l'utilitaire **`xsadmin`***

Arguments de ligne de commande	Nom de la propriété équivalente dans le fichier de propriétés	Description et valeurs valides
-bp	n/a	Indique le port d'écoute. **Valeur par défaut :**`2809`
-ch	n/a	Indique le nom d'hôte JMX pour le serveur de catalogue. **Valeur par défaut :**`localhost`
-clear	n/a	Efface la mappe définie. **Permet d'utiliser les filtres suivants :** `-fm`
-containers	n/a	Pour chaque grille de données et chaque mappe définies, affiche une liste des serveurs de conteneurs. **>Permet d'utiliser les filtres suivants :** `-fnp`
-continuous	n/a	Spécifiez cet indicateur si vous voulez des résultats de taille de mappe en continu pour surveiller la grille de données. Lorsque vous exécutez cette commande avec l'argument **`-mapsizes`** , la taille de la mappe s'affiche toutes les 20 secondes.
-coregroups	n/a	Affiche tous les groupes centraux pour le serveur de catalogue. Cet argument est utilisé pour des diagnostics avancés.
-dismissLink *<domaine_service_catalogue>*	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-dmgr	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-empties	n/a	Spécifiez cet indicateur si vous voulez afficher les conteneurs vides dans les résultats.
-establishLink *<nom_domaine_étranger>*	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower

<hôte1:port1, hôte2:port2...>		XC10 Appliance.
-fc	n/a	Filtre pour ce conteneur uniquement. Si vous effectuez le filtrage des serveurs de conteneurs dans un environnement WebSphere Application Server Network Deployment, utilisez la syntaxe suivante : `<nom_cellule>/<nom_noeud>/<nomServeur_suffixeConteneur>` **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fh	n/a	Filtre pour cet hôte uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-fm	n/a	Filtre uniquement cette mappe. **Utilisez les arguments suivants : `-clear`, `-mapsizes`**
-fnp	n/a	Filtre les serveurs qui n'ont pas de fragments principaux. **Utilisez les arguments suivants : `-containers`**
-fp	n/a	Filtre pour cette partition uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-fs	n/a	Filtre pour ce serveur uniquement. Si vous effectuez le filtrage des serveurs d'applications dans un environnement WebSphere Application Server Network Deployment , utilisez la syntaxe suivante : `<nom_cellule>/<nom_noeud>/<nom_serveur>` **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fst	n/a	Filtre pour ce type de fragment uniquement. Spécifiez P pour les fragments principaux uniquement, A pour les fragments de réplique asynchrone uniquement et S pour les fragments de réplique synchrone uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fz	n/a	Filtre pour cette zone uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-force	n/a	Force l'action qui est dans la commande, en

		désactivant toutes les invites préalables. Cet argument est utile pour exécuter des commandes par lot.
-g	n/a	Spécifie le nom d'ObjectGrid.
-getstatsspec	n/a	Affiche la spécification de statistique actuelle. Vous pouvez définir la spécification de statistique avec l'argument **-setstatsspec**. **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-getTraceSpec	n/a	Affiche la spécification de trace actuelle. Vous pouvez définir la spécification de trace avec l'argument **-settracespec**. **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-h	n/a	Affiche l'aide de l'utilitaire **xsadmin** qui inclut une liste d'arguments.
-hosts	n/a	Affiche tous les hôtes de la configuration.
-jmxUrl	XSADMIN_JMX_URL	Spécifie l'adresse d'un serveur de connecteur d'API JMX au format suivant : `service:jmx:`*`protocole`*`:`*`sap`*. Les définitions des variables *`protocole`* et *`sap`* sont les suivantes : ***`protocole`*** Spécifie le protocole de transport à utiliser pour la connexion au serveur de connecteur. ***`sap`*** Spécifie l'adresse à laquelle le serveur de connecteur se trouve. Pour plus d'informations sur le format de l'URL du service JMX, voir Classe JMXServiceURL (Java™ 2 Platform SE 5.0).
-l	n/a	Affiche toutes les grilles de données et groupes de mappes connues.
-m	n/a	Spécifie le nom du groupe de mappes.
-mapsizes	n/a	Affiche la taille de chaque mappe sur le serveur de catalogue pour vérifier que la distribution des clés est uniforme sur les fragments. **Permet d'utiliser les filtres suivants :** `-fm -fst -fc -fz -fs -fh -fp`
-mbeanservers	n/a	Affiche une liste de tous les noeuds finals de serveur de bean géré.
-overridequorum	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-password	XSADMIN_PASSWORD	Spécifie le mot de passe pour se connecter à l'utilitaire **xsadmin**. Ne spécifiez pas le mot de passe dans votre fichier de propriétés si vous voulez que ce mot de passe reste sécurisé.
-p	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
	n/a	Affiche le placement configuré et le placement de l'environnement d'exécution de votre configuration. Vous pouvez définir la portée de la sortie à une combinaison de grilles de données et de groupes de

		mappes, ou à la configuration entière :
		• Configuration entière :
		`-placementStatus`
		• Pour une grille de données spécifique :
		`-placementStatus -g ma_grille`
		• Pour un groupe de mappes spécifique :
		`-placementStatus -m mon_groupe_de_mappes`
		• Pour une grille de données et un groupe de mappes spécifiques :
		`-placementStatus -g ma_grille -m mon_groupe_de_mappes`

-primaries	n/a	Affiche une liste des fragments primaires.
-profile	n/a	Spécifie un chemin complet au fichier de propriétés pour l'utilitaire **xsadmin**.
-quorumstatus	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-releaseShard *<nom_serveur_conteneur>* *<nom_objectgrid>* *<nom_groupe_mappes>* *<nom_partition>*	n/a	Utilisé en association avec l'argument **-reserveShard**. L'argument **-releaseShard** doit être appelé après qu'un fragment a été réservé et placé. L'argument **-releaseShard** appelle la méthode ContainerMBean.release().
-reserved	n/a	Utilisé avec l'argument **-containers** pour afficher uniquement les fragments qui ont été réservés avec l'argument **-reserveShard**.
-reserveShard *<nom_serveur_conteneur>* *<nom_objectgrid>* *<nom_groupe_mappes>* *<nom_partition>*	n/a	Transfère un fragment primaire vers le serveur de conteneur défini. La méthode ContainerMBean.reserve() est appelée par cet argument.
-resumeBalancing *<nom_objectgrid>* *<nom_ensemble_mappes>*	n/a	Tente d'équilibrer les demandes. Permet des tentatives de rééquilibrage ultérieures sur l'ObjectGrid et le groupe de mappes définis.
-revisions	n/a	Affiche les identificateurs des révisions d'un domaine de service de catalogue avec chaque grille de données, le numéro de partition, le type de partition (principale ou réplique), le domaine de service de catalogue, l'ID de durée de vie et le nombre de révisions des données de chaque fragment. Vous pouvez utiliser cet argument pour déterminer si une réplique asynchrone ou un domaine lié sont interceptés. Cet argument appelle

		la méthode ObjectGridMBean.getKnownRevisions(). **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-routetable	n/a	Affiche l'état actuel de la grille de données à partir d'une perspective serveur client. La table de routage est l'information qu'un serveur client ObjectGrid utilise pour communiquer avec la grille de données. Utilisez la table de routage sous la forme d'une aide au diagnostic lorsque vous tentez d'identifier les problèmes de connexion ou les exceptions `TargetNotAvailable`. **Arguments requis :** Dans un environnement autonome, vous devez spécifier les paramètres **-bp** et **-p** les avec cet argument si vous n'utilisez pas les valeurs par défaut pour le port d'écoute d'amorçage et le port JMX pour l'hôte du serveur de catalogue. **Permet d'utiliser les filtres suivants :** `-fz -fh -fp`
-settracespec *<chaîne_trace>*	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance. Voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance pour plus d'informations sur la configuration de la trace sur le dispositif. **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-swapShardWithPrimary *<nom_serveur_conteneur>* *<nom_objectgrid>* *<nom_ensemble_mappes>* *<nom_partition>*	n/a	Permute le fragment de réplique défini du serveur de conteneur indiqué et le fragment primaire. Cette commande permet d'équilibrer manuellement les fragments primaires lorsque cela est nécessaire.
-setstatsspec *<spécification_stats>*	n/a	Active la collecte des statistiques. Cet argument appelle les méthodes DynamicServerMBean.setStatsSpec et DynamicServerMBean.getStatsSpec. **Permet d'utiliser les filtres suivants :** `-fm -fst -fc -fz -fs -fh -fp`
-suspendBalancing *<nom_objectgrid>* *<nom_ensemble_mappes>*	n/a	Empêche les tentatives d'équilibrage de l'ObjectGrid et du groupe de mappes définis.
-ssl	n/a	Indique que SSL (Secure Sockets Layer) est activé.
-teardown	n/a	**Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp` **Format pour spécifier une liste de serveurs :**

		Format pour spécifier une liste de serveurs :
		`nom_serveur_1,nom_serveur_2 ...`
		Pour arrêter tous les serveurs dans une zone, incluez l'argument -fz :
		`–fz <nom_zone>`
		Pour arrêter tous les serveurs sur un hôte, incluez l'argument -fh :
		`–fh <nom_hôte>`
		Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.

	n/a	Force le placement de fragment à s'exécuter, en ignorant la valeur **numInitialContainers** définie dans le fichier de déploiement XML. Vous pouvez utiliser cet argument lorsque vous effectuez des opérations de maintenance pour pouvoir continuer à placer les fragments, même si la valeur **numInitialContainers** est inférieure à la valeur définie.
-trustPass	XSADMIN_TRUST_PASS	Spécifie le mot de passe pour le fichier de clés certifiées spécifié.
-trustPath	XSADMIN_TRUST_PATH	Spécifie un chemin vers le fichier de clés certifiées. Exemple : `etc/test/security/server.public`
-trustType	XSADMIN_TRUST_TYPE	Spécifie le type de fichier de clés certifiées. Les valeurs valides sont : JKS, JCEK, PKCS12, etc.
-unassigned	n/a	Affiche une liste de fragments qui ne peuvent pas être placés sur la grille de données. Les fragments ne peuvent pas être placés lorsque le service de placement a une contrainte qui empêche le placement.
-username	XSADMIN_USERNAME	Spécifie le nom d'utilisateur pour se connecter à l'utilitaire **xsadmin**.
-v	n/a	Active l'action de ligne de commande prolixe. Utilisez cet indicateur si vous utilisez des variables d'environnement, un fichier de propriétés, ou les deux pour spécifier certains arguments de ligne de commande, et si vous voulez afficher leur valeur.
-xml	n/a	Affiche la sortie filtrée à partir de la méthode PlacementServiceMBean.listObjectGridPlacement(). Les autres arguments **xsadmin** filtrent la sortie de cette méthode et organisent les données dans un format plus exploitable.

Rubrique parent : Surveillance avec l'utilitaire xsadmin

Migration de l'outil `xsadmin` vers l'outil `xscmd`

Dans les versions précédentes, l'outil **`xsadmin`** était un exemple d'utilitaire de ligne de commande pour surveiller l'état de l'environnement. L'outil **`xscmd`** a été introduit comme outil officiel de ligne de commande d'administration et de surveillance. Si vous utilisiez l'outil **`xsadmin`**, faites migrer les commandes vers le nouvel outil **`xscmd`**.

Equivalents des commandes `xsadmin` et `xscmd`

2.5+ **Important :** L'utilitaire **`xsadmin`** est obsolète. Utilisez l'utilitaire **`xscmd`** à la place. L'utilitaire **`xscmd`** est fourni comme utilitaire pris en charge pour la surveillance et l'administration de votre environnement. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

*Tableau 1. Arguments de l'utilitaire **`xsadmin`** et commandes équivalentes **`xscmd`**. Certaines commandes **`xscmd`** ont une f forme longue. La forme courte des commandes a un tiret (-) et la forme longue, deux (--). Vous pouvez utiliser l'une à la inversement.*

Arguments de ligne de commande `xsadmin`	**Commande équivalente `xscmd`**	**Paramètres de commande `xscmd`**
`-bp`	• **`-cep`** *hostname*:*listener_port* • **`--catalogEndpoint`** *hostname*:*listener_port*	n/a
`-ch`	• **`-cep`** *hostname*:*listener_port* • **`--catalogEndpoint`** *hostname*:*listener_port*	n/a
`-clear`	**`-c clearGrid`**	**`-g, -ms, -v, -m, (-cep)`**
`-containers`	• **`-c showPlacement -container`** *containerName* • **`-c showPlacement -server`** *serverName*	**`-e, -i, , -st, -snp, -ct, -s, -p, -hf`**
`-continuous`	n/a	n/a
`-coregroups`	• • **`-c listCoreGroupMembers -cg`** *core_group*	n/a
`-dismissLink` *<catalog_service_domain>*	**`-c dismissLink`**	• **`-fd`** *<foreignCatalogServiceDomain* • **`--foreignCatalogServiceDomain`** *<foreignCatalogServiceDomain>*
`-dmgr`	s/o - Cet argument est déterminé automatiquement avec **`xscmd`**	n/a
`-empties`	arg spécifique d'une nouvelle commande	n/a
`-establishLink` *<foreign_domain_name> <host1:port1,host2:port2...>*	**`-c establishLink`**	• **`-fd`** *<foreignCatalogServiceDomain <host1:port1,host2:port2...>* • **`--foreignCatalogServiceDomain`** *<foreignCatalogServiceDomain>* **`-1`** *<host1:port1,host2:port2...>*
`-fc`	• **`-ct`** • **`--container`**	n/a
`-fh`	• **`-hf`** • **`--hostFilter`**	n/a
`-fm`	• **`-m`** • **`--map`**	n/a
`-fnp`	• **`-snp`** • **`--serversWithNoPrimaries`**	n/a
`-fp`	• **`-p`** • **`--partitionId`**	n/a

-fs	• **-s** • **--server**	n/a
-fst	• **-st** <*shard_type*> • **--shardType** <*shard_type*> Shard values: P=primary A=asyncReplica S=syncReplica	n/a
-fz	• **-z** • **--zone**	n/a
-force	arg spécifique d'une nouvelle commande	
-g	• **-g** • **--objectGrid**	n/a
-getstatsspec	**-c getStatsSpec**	n/a
-getTraceSpec	**-c getTraceSpec**	n/a
-h	Vous pouvez exécuter l'aide avec ou sans un nom de commande spécifique : • **-h** • **--help** • **-h** <*command_name*> • **--help** <*command_name*>	n/a
-hosts	**-c listHosts**	**-g, -ms, -st, -c, -s, -hf, -z**
-jmxUrl	• **-cep** *hostname*:*listener_port* • **--catalogEndpoint** *hostname*:*listener_port*	n/a
-l	**-c listObjectGridNames**	n/a
-m	• **-ms** • **--mapSet**	n/a
-mapsizes	**-c showMapSizes**	**-g, -ms, -i, [-ct, -z, -s, -hf, sht [**
-mbeanservers	**-c listAllJMXAddresses**	n/a
-overridequorum	**-c overrideQuorum**	n/a
-password	• **-pwd** • **--password**	n/a
-p	• **-cep** *hostname*:*listener_port* • **--catalogEndpoint** *hostname*:*listener_port*	n/a
-placementStatus	**-c placementServiceStatus**	**-g, -ms**
-primaries	**-c showPlacement -sf P**	**-e, -i, , -st, -snp, -ct, -s, -p, -hf**
-profile	Pour enregistrer les paramètres de sécurité actuels dans un profil de sécurité : • **-ssp** *profile_name* • **--saveSecProfile** *profile_name* Pour utiliser un profil de sécurité spécifié : • **-sp** *profile_name* • **--securityProfile** *profile_name*	
-quorumstatus	**-c showQuorumStatus**	n/a
-releaseShard	**-c releaseShard**	**-c, -g, -ms, -p**

Avertissement : Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	
-reserved	• **-sf R** • **--shardFilter R**	n/a
-reserveShard **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c reserveShard** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c, -g, -ms, -p**
-resumeBalancing *<objectgrid_name>* *<map_set_name>* **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c resumeBalancing** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-g, -ms**
-revisions	**-c revisions**	**-s,-p,-g,-m**
-routetable	**-c routetable**	**-z, -hf,-p,-g,-ms**
-settracespec *<trace_string>*	**-c setTraceSpec**	**-spec** *<trace_string>*
-swapShardWithPrimary **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c swapShardWithPrimary** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c -g, -ms, -p**
-setstatsspec *<stats_spec>*	**-c setStatsSpec**	**-spec** *<stats_spec>*
-suspendBalancing *<objectgrid_name>* *<map_set_name>* **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c suspendBalancing** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-g, -ms**
-ssl	• **-ssl** • **--enableSSL**	n/a
-teardown **Avertissement :** Cet argument de ligne de	**-c teardown** **Avertissement :** Cet argument de ligne de commande ne s'applique pas	**-f, , -st, -snp, -c, -s, -p, -hf, -z,**

commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	pour les configurations WebSphere DataPower XC10 Appliance.	
-triggerPlacement **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c triggerPlacement** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-g, -ms**
-trustPass	• **-tsp** • **--trustStorePassword**	n/a
-trustPath	• **-ts** • **--trustStore**	n/a
-trustType	• **-tst** • **--trustStoreType**	n/a
-unassigned	**-c showPlacement -sf U**	**-e, -i, , -st, -snp, -ct, -s, -p, -hf**
-username	• **-user** • **--username**	n/a
-v	• **-v** • **--verbose**	n/a
-xml	**-c showPlacement**	n/a

Rubrique parent : Surveillance avec l'utilitaire xsadmin

Surveillance de la santé de l'environnement

2.5+ Message Center fournit une vue agrégée des notifications d'événements pour les messages de journal et de l'outil de diagnostic de premier niveau. Vous pouvez afficher ces notifications d'événements avec : Message Center dans la console Web, l'utilitaire **xscmd**, les fichiers journaux de santé, l'interface de commande HTTP, ou un programme avec MBeans.

2.5+ **Surveillance de l'état de santé du système - présentation**
Message Center agrège en temps réel les événements d'état de santé provenant de tous les serveurs de conteneur et de catalogue d'une collectivité. Lorsque Message Center est configuré, vous pouvez afficher une présentation des événements critiques qui surviennent sur différents serveurs sans collecter les journaux de chaque serveur.

2.5+ **Affichage des notifications d'événement de santé dans Message Center**
Vous pouvez utiliser le centre de messages (Message Center) de la console Web pour évaluer en temps réel la santé de l'intégralité de la grille de données et de la collectivité. Les événements qui s'affichent dans Message Center sont un sous-ensemble d'événements qui sont filtrés pour afficher les problèmes les plus critiques.

2.5+ **Affichage des informations de santé avec l'utilitaire xscmd**
Vous pouvez afficher les notifications d'événements en cours, l'historique de notification d'événement et afficher et définir des filtres de notification à partir du centre de message (Message Center) avec l'utilitaire **xscmd**.

Que faire ensuite

Vous pouvez afficher un récapitulatif de l'état de santé global du matériel et des logiciels du dispositif. Utilisez pour cela l'une des options suivantes :

- Affichez et téléchargez le fichier journal de santé. Pour plus d'informations, voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance.
- Utilisez la commande **GetHealthStatus** dans l'interface de commande HTTP. Pour plus d'informations, voir Administration à l'aide de l'interface de commande HTTP et Référence de l'interface de commande HTTP.

Rubrique parent : Surveillance

Surveillance de l'état de santé du système - présentation

2.5+ Message Center agrège en temps réel les événements d'état de santé provenant de tous les serveurs de conteneur et de catalogue d'une collectivité. Lorsque Message Center est configuré, vous pouvez afficher une présentation des événements critiques qui surviennent sur différents serveurs sans collecter les journaux de chaque serveur.

Mise en oeuvre de Message Center

Le centre de messages Message Center est activé par défaut. Vous pouvez le désactiver dans l'interface utilisateur.

Les déploiements de grille de données peuvent impliquer des dizaines, voire des centaines, de processus serveur répartis. Si un incident se produit, vous pouvez ouvrir le fichier journal du serveur de conteneur concerné pour analyser le problème en détail.

Message Center est constitué des composants suivants :

Agrégation d'événements

Lorsque vous configurez la surveillance de la santé sur un serveur de catalogue, vous recevez les événements agrégés qui affectent la santé de l'ensemble du domaine de service de catalogue. Les informations reçues incluent la source et la gravité des types d'événement suivants :

- Tous les événements de l'outil de diagnostic de premier niveau
- Toutes les entrées de journal WARNING ou SEVERE
- La liste de toutes les entrées de journal, notamment INFO, WARNING et SEVERE
- Les opérations de démarrage et d'arrêt du serveur
- La grille de données approche de sa limite de capacité
- Perte ou récupération du quorum
- Déclenchement des alertes SNMP activées
- Retard dans la réplication pendant une période de 15 minutes

Message Center dans la console Web

Message Center dans la console Web affiche les enregistrements d'événement agrégés. Ces événements incluent les événements récents et les notifications de mise à jour en temps réel qui se produisent après l'ouverture de la console.

Evénements dans l'utilitaire `xscmd`

Vous pouvez également afficher la liste des événements récents à l'aide de l'utilitaire **`xscmd`**. Lorsque des événements se produisent, vous pouvez rediriger les enregistrements d'événement pour créer des utilitaires de scriptage automatique.

Beans géré pour l'intégration dans d'autres logiciels de surveillance

Vous pouvez également utiliser les beans gérés de gestion (MBeans) disponibles pour connecter Message Center à vos autres logiciels de surveillance JMX (Java Management Extensions). La documentation de ces beans gérés est incluse dans la documentation des API.

2.5+

Récapitulatif de l'état de santé

Les journaux de santé et la commande **`GetHealthStatus`** de l'interface de commande HTTP vous permettent d'obtenir un récapitulatif de l'état de santé du matériel et des logiciels du dispositif. Les informations fournies incluent le placement de la grille de données et son état de réplication, des avertissements concernant le matériel, l'état de santé du système et du réseau, et d'autres messages d'état.

Message Center et Analyseur de journal

L'analyseur de journal est un autre outil qui permet d'analyser un groupe de messages de journal. Cet outil impose de collecter manuellement les journaux des divers serveurs dans votre environnement. Ensuite, vous pouvez exécuter l'outil pour créer des rapports sur les problèmes. Utilisez l'analyseur de journal pour exécuter une analyse post-mortem des journaux lorsque vous devez analyser un nombre de messages supérieur au sous-ensemble de 1000 messages que vous pouvez afficher dans Message Center. Utilisez Message Center pour surveiller en temps réel l'état de la grille de données pour identifier rapidement les problèmes qui surviennent. Ensuite, vous pouvez vérifier les fichiers journaux du serveur de conteneur associé ou utiliser l'analyseur de journal pour analyser le problème plus en détail.

Configuration et architecture de la surveillance de la santé

Message Center est activé par défaut. Vous pouvez le désactiver dans l'interface utilisateur. Lorsque Message Center est activé, les serveurs de catalogue de votre collectivité jouent le rôle de concentrateurs pour la surveillance de l'état de santé. Généralement, les messages de Message Center proviennent du

serveur de catalogue qui réside sur le dispositif à partir duquel vous avez créé la collectivité. Chaque concentrateur dispose de ses propres abonnements et historiques d'événements. Chaque événement de l'historique porte un numéro de séquence. Les historiques d'événements sur les serveurs de catalogue ne sont pas synchronisés et sont différents. Les serveurs de catalogue peuvent s'abonner aux événements de journal et de l'outil de diagnostic de premier niveau des autres serveurs de catalogue.

Rubrique parent : 2.5+ Surveillance de la santé de l'environnement

Affichage des notifications d'événement de santé dans Message Center

Vous pouvez utiliser le centre de messages (Message Center) de la console Web pour évaluer en temps réel la santé de l'intégralité de la grille de données et de la collectivité. Les événements qui s'affichent dans Message Center sont un sous-ensemble d'événements qui sont filtrés pour afficher les problèmes les plus critiques.

Procédure

- Affichez les erreurs graves, les messages de l'outil de diagnostic de premier niveau et démarrez et arrêtez les événements serveur via les notifications d'événements dans la console Web. Ces notifications s'affichent automatiquement lorsque vous êtes connecté à une page dans la console Web.
- Affichez les messages dans Message Center. Dans la console Web, cliquez sur **Surveillance** > **Message Center**. Message Center affiche les 1000 derniers messages critiques qui ont été envoyés via le concentrateur de messages du serveur de catalogue.

  Le concentrateur de messages du serveur de catalogue filtre les messages qui s'affichent et il affiche donc les 1000 messages. Par conséquent, Message Center contient un sous-ensemble de tous les événements critiques qui surviennent sur les serveurs de catalogue. Si de nouveaux messages deviennent disponibles alors que la page est ouverte, un message d'information comportant une option d'actualisation des informations s'affiche en haut de la page.
- Filtrez les messages affichés dans Message Center. Vous pouvez ajouter jusqu'à trois règles de filtrage. Une règle est constituée d'une colonne, d'une condition et d'une valeur.
  1. Dans la console Web, cliquez sur **Surveillance** > **Message Center**.
  2. Cliquez sur le bouton de filtrage ( ).
  3. Ajoutez une règle.
     a. Cliquez sur le bouton d'ajout ( ).
     b. Dans Message Center, choisissez la colonne à filtrer :

        ID

        ID d'événement généré par Message Center.

        Type

        Le type de message qui indique la gravité du message. Les valeurs admises sont : Grave, Avertissement, Erreur et Information.

        Date

        Date et heure de génération du message.

        Source

        Serveur d'origine du message.

        Message

        Texte du message de l'événement de message.

     c. Sélectionnez la condition à laquelle vous voulez appliquer le filtre. La liste suivante de conditions s'appliquent à pratiquement toutes les colonnes, à l'exception de la colonne de date et de type :
        - Contient
        - Est
        - Commence par
        - Se termine par

     d. Entrez la valeur de filtrage de la colonne.

     Exemple : pour n'afficher que les messages de server1, sélectionnez la colonne **Source**. Sélectionnez la condition **Est**. Pour la valeur, tapez `server1`.
  4. Vous pouvez choisir d'établir une correspondance avec certaines des règles que vous avez définies ou toutes les règles.
  5. Cliquez sur **Filtrer** pour appliquer les filtres définis à la sortie de Message Center.
- 2.5+ Désactivez Message Center. La désactivation de Message Center interrompt la détection de nouveaux messages. Cliquez sur **Désactiver Message Center et les alertes en incrustation**. Cela fait, appliquez la modification.

Que faire ensuite

Vous pouvez exécuter une analyse supplémentaire sur vos fichiers journaux. Pour plus d'informations, voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance.

Rubrique parent : 2.5+ Surveillance de la santé de l'environnement

Affichage des informations de santé avec l'utilitaire `xscmd`

2.5+ Vous pouvez afficher les notifications d'événements en cours, l'historique de notification d'événement et afficher et définir des filtres de notification à partir du centre de message (Message Center) avec l'utilitaire **`xscmd`**.

Avant de commencer

- Démarrez l'utilitaire **`xscmd`** et connectez-le au domaine de service de catalogue. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Procédure

- Affichez l'historique des notifications d'événements avec l'utilitaire **`xscmd`**. La sortie apparaît dans un tableau.

```
xscmd -c showNotificationHistory -cep nom_hôte:port(,nom_hôte:port)
```

- Ecoutez les nouvelles notifications.

```
xscmd -c listenForNotifications -cep nom_hôte:port(,nom_hôte:port)
```

  La sortie apparaît dans un format brut et s'exécute jusqu'à ce que vous arrêtiez la commande. Vous pouvez écrire des scripts supplémentaires pour analyser la sortie.

- Activez le filtrage pour toutes les entrées de journal à venir, y compris INFO, WARNING et SEVERE. Par défaut, Message Center et les commandes affichent uniquement les erreurs WARNING et SEVERE et les événements. Vous pouvez définir le filtre pour tous les serveurs dans l'environnement ou sur un serveur unique. Ce paramétrage n'affecte que les résultats à venir.

```
xscmd -c setNotificationFilter -fs <expression régulière>  [-server <nom_serveur>]
```

- Affichez les filtres de notification en cours pour tous les serveurs de l'environnement ou un serveur unique.

```
xscmd -c getNotificationFilter [-s nom_serveur]
```

Rubrique parent : 2.5+ Surveillance de la santé de l'environnement

Surveillance avec SNMP (Simple Network Monitoring Protocol)

Avec le support SNMP (Simple Network Monitoring Protocol), vous pouvez contrôler l'état d'un dispositif IBM® WebSphere DataPower XC10 Appliance dans le cadre d'un grand groupe de systèmes dans un centre de données. La surveillance SNMP améliore votre capacité de notifier les problèmes et de les résoudre rapidement.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez utiliser le contrôle que si un client est configuré.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Pourquoi et quand exécuter cette tâche

Procédez comme suit pour activer la surveillance SNMP pour votre IBM WebSphere DataPower XC10 Appliance. Les paramètres SNMP sont uniques pour le dispositif et ne sont pas propagés auprès des autres dispositifs de la collectivité.

A faire : Les statistiques SNMP étant fournies pour un seul dispositif, les valeurs rapportées sont différentes des valeurs qui sont rapportées par les panneaux de surveillance dans l'interface utilisateur. L'interface utilisateur rapporte des données sur la collectivité. Par exemple, une valeur de la table gridStats, de la table mapStats ou de la table jvmStats SNMP représente un sous-ensemble de la valeur totale rapportée dans l'interface utilisateur.

1. Activation de la surveillance SNMP du dispositif
   La surveillance SNMP (Simple Network Monitoring Protocol) peut être activée pour le dispositif IBM WebSphere DataPower XC10 Appliance. Vous pouvez télécharger les bases d'informations de gestion (MIB) fournies dans le dispositif pour définir les données SNMP disponibles pour le client SNMP.
2. Configuration de communautés SNMP
   Vous pouvez définir l'accès aux données SNMP (Simple Network Management Protocol) sur le dispositif en créant une ou plusieurs communautés SNMP. Une communauté SNMP est nécessaire lorsque le contrôle est activé.
3. Configuration des abonnés aux interceptions SNMP
   Les abonnés aux interceptions représentent les clients SNMP (Network Monitoring Protocol) utilisés pour communiquer avec l'agent SNMP intégré au dispositif.
4. Gestion des abonnements aux interceptions SNMP
   Une interruption SNMP (Simple Network Monitoring Protocol) est une notification d'événement ou d'état générée par l'agent SNMP intégré au dispositif. En utilisant des abonnements aux interceptions SNMP, vous définissez les interceptions SNMP que l'agent communique aux clients SNMP.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Rubrique parent : Surveillance

Activation de la surveillance SNMP du dispositif

La surveillance SNMP (Simple Network Monitoring Protocol) peut être activée pour le dispositif IBM® WebSphere DataPower XC10 Appliance. Vous pouvez télécharger les bases d'informations de gestion (MIB) fournies dans le dispositif pour définir les données SNMP disponibles pour le client SNMP.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez utiliser le contrôle que si un client est configuré.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Pourquoi et quand exécuter cette tâche

Par défaut, la surveillance SNMP n'est pas activée pour IBM WebSphere DataPower XC10 Appliance. Les paramètres SNMP sont uniques pour le dispositif et ne sont pas propagés auprès des autres dispositifs de la collectivité.

Procédure

1. Accédez au panneau de surveillance. Dans l'interface utilisateur de IBM WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres SNMP**.
2. Cochez la case pour activer la surveillance SNMP. Si vous voulez la désactiver, désélectionnez la case.
3. Téléchargez les fichiers MIB (Management Information Base) qui sont disponibles sur le dispositif.

   Les fichiers Enterprise MIB décrivent les fonctions et les données disponibles auprès de l'agent SNMP imbriqué pour que votre client puisse y accéder de façon appropriée. Le client peut lancer les commandes SNMP `GET`, `GET-NEXT` et `GET-BULK`. Vous pouvez télécharger ces fichiers MIB et les importer vers le client pour accéder aux données en plus des définitions de données MIB-II. Développez les **MIB d'entreprise** et cliquez sur le nom de chaque base d'informations de gestion à partir du dispositif.

   Statistiques MIB

   Les statistiques MIB comprennent des informations similaires aux statistiques que vous pouvez voir avec la fonction de surveillance de l'interface utilisateur. La base d'informations de gestion MIB contient aussi des statistiques sur la machine virtuelle Java™.

   MIB statut matériel

   Le MIB statut matériel comprend des informations sur le statut du matériel : les températures, la date et l'heure, etc.

   MIB notifications matériel

   La **`MIB notifications matériel`** permet de définir les informations disponibles pour les commandes `SNMP TRAP`.

Résultats

A l'issue de ces étapes, vous avez activé la surveillance SNMP pour le dispositif et téléchargés les données MIB pour le client SNMP.

Que faire ensuite

Utilisez votre client SNMP pour afficher les données MIB.

A faire : Les statistiques SNMP étant fournies pour un seul dispositif, les valeurs rapportées sont différentes des valeurs qui sont rapportées par les panneaux de surveillance dans l'interface utilisateur. L'interface utilisateur rapporte des données sur la collectivité. Par exemple, une valeur de la table gridStats, de la table mapStats ou de la table jvmStats des statistiques SNMP est un sous-ensemble de la valeur totale rapportée dans l'interface utilisateur.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique suivante : Configuration de communautés SNMP

Configuration de communautés SNMP

Vous pouvez définir l'accès aux données SNMP (Simple Network Management Protocol) sur le dispositif en créant une ou plusieurs communautés SNMP. Une communauté SNMP est nécessaire lorsque le contrôle est activé.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez utiliser le contrôle que si un client est configuré. WebSphere DataPower XC10 Appliance prend en charge SNMP version v2c.

Pourquoi et quand exécuter cette tâche

Une communauté est nécessaire pour s'authentifier auprès de l'agent SNMP intégré et accéder aux données SNMP. L'agent SNMP intégré au dispositif attend que le client fournisse un nom de communauté défini pour s'authentifier auprès de l'agent et retourner les données demandées. Si aucune communauté n'est fournie, l'agent SNMP ignore la demande du client. Les communautés SNMP sont uniques pour le dispositif et ne sont pas propagées auprès des autres dispositifs de la collectivité.

Procédure

1. Accédez au panneau de surveillance. Dans la barre de menus dans la partie supérieure de l'interface utilisateur de IBM® WebSphere DataPower XC10 Appliance, accédez à **Dispositif** > **Surveillance SNMP**.
2. Développez **Communautés SNMP**.
3. Cliquez sur **Créer une communauté**.
4. Complétez le formulaire d'abonnement à la communauté SNMP à créer.

   `Nom`

   Nom décrivant une communauté SNMP.

   `Restriction hôte`

   Cette zone spécifie une adresse IP à utiliser pour la communication aux formats IPv4 et IPv6. Elle accepte également les noms d'hôte résolus en hôte. Vous pouvez limiter ultérieurement l'accès à un sous-réseau spécifié à l'aide d'une adresse IP ou d'un nom d'hôte de routage CIDR (Classless Inter-Domain Routing). Si une restriction d'hôte est incluse, les communications avec une adresse adresse IP ou un autre sous-réseau sont refusées. Si vous laissez cette zone vide, vous autorisez la communication avec toutes les adresses IP.

5. Cliquez sur l'icône de suppression ( ) pour supprimer une communauté SNMP. Vous ne pouvez pas modifier les communautés existantes. Si vous devez modifier une communauté, vous devez la supprimer et la recréer.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique précédente : Activation de la surveillance SNMP du dispositif
Rubrique suivante : Configuration des abonnés aux interceptions SNMP

Configuration des abonnés aux interceptions SNMP

Les abonnés aux interceptions représentent les clients SNMP (Network Monitoring Protocol) utilisés pour communiquer avec l'agent SNMP intégré au dispositif.

Avant de commencer

Les abonnés aux interceptions SNMP doivent être des clients SNMP existants. Configurez le client SNMP pour recevoir des interceptions avant de créer l'abonné aux interceptions dans le dispositif IBM® WebSphere DataPower XC10 Appliance interface utilisateur.

Pourquoi et quand exécuter cette tâche

Un abonné aux interceptions SNMP est requis pour l'acheminement des notifications relatives au dispositif à un client SNMP abonné. Lors de la création d'un abonné aux interceptions, vous devez fournir des informations sur les clients SNMP qui reçoivent des interceptions SNMP. Les interceptions décrivent les situations nécessitant l'attention de l'administrateur d'IBM WebSphere DataPower XC10 Appliance. L'agent SNMP n'envoie pas d'interceptions aux clients qui ne sont pas définis comme abonnés.

Tous les abonnés aux interceptions SNMP reçoivent automatiquement des interceptions relatives aux limites de stockage de la grille de données et aux limites de stockage du dispositif. Pour plus d'informations sur les interceptions spécifiques pouvant être définies, voir Référence d'interruption SNMP.

Procédure

1. Accédez au panneau Paramètres SNMP. Dans l'interface utilisateur de WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres SNMP**.
2. Développez **Abonnés aux interceptions**.
3. Cliquez sur **Créer un abonné aux interceptions**.
4. Complétez le formulaire pour décrire les abonnés aux interceptions SNMP à créer.

   `Hôte`

   Définit l'adresse IP sur laquelle le client SNMP écoute les informations d'interception.

   `Numéro de port du client`

   Définit le port sur lequel le client SNMP écoute les informations d'interception.

   `Communauté`

   Définit une communauté SNMP dont le client est membre.

5. Cliquez sur l'icône de suppression ( ) pour supprimer un abonné aux interceptions SNMP. Vous ne pouvez pas modifier les abonnés existants. Si vous devez modifier un abonné, vous devez le supprimer et le recréer.

Que faire ensuite

Après avoir défini au moins un abonné aux interceptions SNMP qui peut accéder au dispositif, vous pouvez configurer les interceptions que doit communiquer l'agent SNMP.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique précédente : Configuration de communautés SNMP
Rubrique suivante : Gestion des abonnements aux interceptions SNMP

Gestion des abonnements aux interceptions SNMP

Une interruption SNMP (Simple Network Monitoring Protocol) est une notification d'événement ou d'état générée par l'agent SNMP intégré au dispositif. En utilisant des abonnements aux interceptions SNMP, vous définissez les interceptions SNMP que l'agent communique aux clients SNMP.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez pas recevoir les interceptions SNMP sans client SNMP valide configuré.

Pourquoi et quand exécuter cette tâche

Pour définir les interceptions à communiquer, procédez comme suit.

Procédure

1. Accédez au panneau Paramètres SNMP. Dans la barre de menus dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, accédez à **Dispositif** > **Paramètres SNMP** > **Abonnements aux interceptions**.
2. Sélectionnez ou désélectionnez des interceptions individuelles. Vous pouvez sélectionnez les interceptions SNMP en sélectionnant et en effaçant des interceptions individuelles. Tous les abonnés aux interceptions SNMP reçoivent automatiquement des interceptions relatives aux limites de stockage de la grille de données et aux limites de stockage du dispositif. Pour plus d'informations sur les interceptions spécifiques pouvant être définies, voir Référence d'interruption SNMP.

Résultats

A l'issue de ces étapes, vous avez défini le groupe d'interceptions SNMP signalées aux clients SNMP abonnés.

Référence d'interruption SNMP
Les événements ci-dessous peuvent être interceptés par l'agent SNMP (IBM WebSphere DataPower XC10 Appliance Simple Network Management Protocol) et communiqués aux clients SNMP abonnés.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique précédente : Configuration des abonnés aux interceptions SNMP

Référence d'interruption SNMP

Les événements ci-dessous peuvent être interceptés par l'agent SNMP (IBM® WebSphere DataPower XC10 Appliance Simple Network Management Protocol) et communiqués aux clients SNMP abonnés.

Interceptions SNMP disponibles pour contrôler votre dispositif IBM WebSphere DataPower XC10 Appliance

Tableau 1. Interceptions SNMP de IBM WebSphere DataPower XC10 Appliance.

Les interceptions ci-dessous sont configurées par défaut. Vous ne pouvez pas les désactiver.

Description
Cette notification indique que la grille de données XC10 spécifique a atteint 60 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifiée est repassée en dessous de 60 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifique a atteint 80 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifiée est repassée en dessous de 80 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifique a atteint sa capacité.
Cette notification indique que la grille de données XC10 spécifiée est repassée en dessous de sa capacité.
Cette notification indique que le stockage du dispositif XC10 a atteint 80 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 est repassé en dessous de 80 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 a atteint 90 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 est repassé en dessous de 90 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 a atteint sa capacité.
Cette notification indique que le stockage du dispositif XC10 est repassé en dessous de sa capacité.

Tableau 2. Interceptions SNMP du dispositif.

Les interceptions ci-dessous sont activées par défaut. Vous pouvez les désactiver dans l'interface utilisateur.

Valeur par défaut	Description
Y	L'interface Ethernet passe à l'état inactif.
Y	L'interface Ethernet est passée à l'état actif.
Y	L'agent SNMP est réinitialisé avec des modifications de configuration potentielles
Y	L'agent SNMP est en cours d'arrêt.
Y	Le capteur du ventilateur est repassé à l'état normal.
Y	Le capteur du ventilateur est passé à un état indésirable.
Y	Le détecteur de tension est repassé à l'état normal.
Y	Le capteur de tension est passé à un état indésirable.
Y	Le détecteur de température est repassé à une plage normale.
Y	Le détecteur de température est passé à une plage indésirable.
Y	Le capteur d'alimentation est repassé à l'état normal.

Y	Le capteur d'alimentation est repassé à l'état normal.
Y	Le capteur d'alimentation est passé à un état indésirable.

Rubrique parent : Gestion des abonnements aux interceptions SNMP

Configuration de la consignation à distance

Vous pouvez activer la consignation à distance pour enregistrer les entrées de journal sur un serveur distant en dehors du dispositif. La consignation à distance peut être utile lorsque vous devez définir un niveau de journal de débogage détaillé pour isoler un problème ou surveiller un comportement sur une longue période.

Avant de commencer

- Vous devez disposer d'un serveur syslog qui écoute les événements et les capture.
- Les noms des serveurs de catalogue, de conteneur et d'applications (si vous utilisez WebSphere Application Server) ne doivent contenir que des caractères alphanumériques. Syslog RFC 1364 n'autorise pas les caractères non alphanumériques pour la zone TAG. La zone TAG contient le nom du serveur dans les messages syslog.

Pourquoi et quand exécuter cette tâche

Utilisez la consignation à distance pour analyser les données d'historique. Le dispositif conserve un nombre limité de fichiers journaux dans le système. Configurez la consignation à distance si vous voulez enregistrer un plus grand nombre de fichiers journaux pour l'analyse. Le serveur de consignation à distance agrège les données de plusieurs dispositifs afin que vous puissiez configurer l'ensemble de la collectivité pour envoyer des fichiers au même serveur de consignation à distance.

Procédure

Configurez un serveur de consignation à distance dans le dispositif interface utilisateur.

a. Dans interface utilisateur, cliquez sur **Dispositif** > **Identification des incidents** > **Journaliation** > **Configurer la consignation à distance**.
b. Entrez l'hôte distant et le port du serveur de journalisation distant.
c. Entrez le seuil de gravité des messages à envoyer au serveur de journalisation distant. Pour envoyer des messages d'avertissement et de gravité, sélectionnez **WARNING**. Pour envoyer des messages graves, sélectionnez **SEVERE**.

Résultats

Les messages sont envoyés au serveur de journalisation distant configuré pour l'archivage et l'analyse.

Rubrique parent : Surveillance

Java

Configuration d'une application de grille de données pour l'utilisation de l'authentification client

Si la grille de données que vous configurez pour l'application utilise la sécurité, vous devez configurer un fichier `client.properties` comportant des paramètres à transmettre à l'application de grille de données.

Avant de commencer

Vous devez disposer d'une application de grille de données qui utilise WebSphere DataPower XC10 Appliance.

Procédure

1. Sécurisez la grille de données utilisée par l'application. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
2. Configurez l'application de grille de données afin de fournir des données d'identification utilisateur. Pour configurer les données d'identification utilisateur, vous devez utiliser un fichier `client.properties`. Vous pouvez utiliser le fichier `sampleClient.properties` qui se trouve dans le répertoire *rép_base_wxs*`/properties` d'une installation WebSphere eXtreme Scale Client pour créer votre fichier de propriétés. Apportez les modifications suivantes au fichier `client.properties` :
   - **securityEnabled** : Affectez à la propriété **securityEnabled** la valeur `true` (valeur par défaut) pour activer la sécurité client, ce qui inclut l'authentification.
   - **credentialAuthentication** : Affectez à la propriété **credentialAuthentication** la valeur `Supported` (valeur par défaut), qui indique que le client prend en charge l'authentification de données d'identification.
   - **transportType** : Affectez à la propriété **transportType** la valeur `TCP/IP`, ce qui signifie qu'aucune couche SSL (Secure Sockets Layer) n'est utilisée.
   - **singleSignOnEnabled** : Affectez à la propriété **singleSignOnEnabled** la valeur `false` (valeur par défaut). La connexion unique (Single sign-on) n'est pas disponible.
   - Les données d'identification d'un utilisateur autorisé à accéder à la grille de données :

     ```
     credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins
     .UserPasswordCredentialGenerator
     credentialGeneratorProps=<nom_utilisateur> <motdepasse>
     ```

     Exemple :

     ```
     credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins
     .UserPasswordCredentialGenerator
     credentialGeneratorProps=xcadmin xcadmin
     ```

   - Pour utiliser un mot de passe codé, vous pouvez faire appel à l'utilitaire **FilePasswordEncoder**. Cet utilitaire se trouve dans le répertoire *rép_base_wxs*`/ObjectGrid/bin`.

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Java

Configuration de TLS pour les applications de grille de données

Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Avant de commencer

- Vous devez disposer d'une application de grille de données qui utilise WebSphere DataPower XC10 Appliance.
- Vous devez disposer des droits d'accès d'administrateur du dispositif.
- Vous devez disposer d'un fichier de clés ou d'un fichier de clés certifiées avec les mots de passe associés à ajouter à la configuration du dispositif. Pour modifier le fichier de clés certifiées existant, vous pouvez le télécharger à partir du dispositif. Vous devez mettre à jour le fichier de clés certifiées à l'aide des certificats publics des clients. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..
- Vous devez avoir accès à l'outil **keytool**. Cet outil se trouve dans le répertoire *rép_base_java*`/bin`.

Pourquoi et quand exécuter cette tâche

Le dispositif doit accréditer les clients qui se connectent à la grille de données. Les paramètres TLS s'appliquent à l'interface utilisateur et aux grilles de données. Les paramètres sont appliqués à tous les dispositifs de la collectivité.

Procédure

1. Téléchargez le fichier de clés certifiées actif. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Paramètres** > **TLS (Transport Layer Security)**. Cliquez sur **Télécharger un fichier de clés certifiées actif** et rappelez-vous l'emplacement dans lequel vous avez enregistré le fichier sur disque, par exemple dans le répertoire `/downloads/trustStore.jks`.
2. Si nécessaire, créez un certificat et exportez le certificat public.
   a. Créez une clé privée dans le fichier de clés. La commande ci-dessous crée le fichier de clés `key.jks` qui contient la clé "ogsample". Ce fichier de clés `key.jks` est utilisé en tant que fichier de clés certifiées SSL. Exécutez la commande suivante :

      ```
       keytool -genkey -alias ogsample -keystore key.jks -storetype JKS -keyalg rsa
      -dname "CN=ogsample, U=Your Organizational Unit, O=Your Organization, L=Your
      City, S=Your State, C=Your Country" storepass ogpass -keypass ogpass -validity
      3650
      ```

   b. Exportez le certificat public. La commande ci-dessous extrait le certificat public de la clé "ogsample" et stocke la clé dans le fichier `temp.key`.

      ```
      keytool -export -alias ogsample -keystore key.jks -file temp.key -storepass
      ogpass
      ```

3. Ajoutez le certificat client au fichier de clés certifiées. Exécutez l'outil **keytool** pour importer le certificat public client vers le fichier de clés certifiées.

   ```
   keytool -import -noprompt -alias "ogsample" -keystore /downloads/trustStore.jks
   -file temp.key -storepass xc10pass -storetype jks
   ```

4. Téléchargez les informations de fichier de clés certifiées vers le dispositif. Dans l'interface utilisateur, cliquez sur **Dispositif** > **Paramètres** > **TLS (Transport Layer Security)**. Téléchargez le fichier `/downloads/trustStore.jks` mis à jour. Cliquez sur **Soumettre les paramètres TLS** pour enregistrer votre configuration.
5. Mettez à jour le fichier `client.properties`. Pour plus d'informations sur l'emplacement de ce fichier et sur les propriétés qu'il contient, voir Fichier de propriétés du client. Définissez les propriétés ci-dessous dans le fichier `client.properties`.

   ```
   securityEnabled=true
   transportType=SSL-Required
   alias=ogsample
   contextProvider=IBMJSSE2
   protocol=TLS
   keyStoreType=JKS
   keyStore=key.jks
   ```

```
keyStorePassword=ogpass
trustStoreType=JKS
trustStore=/downloads/trustStore.jks
trustStorePassword=xc10pass
```

Que faire ensuite

- Redémarrez l'application avec la nouvelle configuration pour vous connecter à l'aide de TLS (Transport Layer Security).

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Traitement des incidents

Pour diagnostiquer et résoudre les problèmes que vous rencontrez dans votre environnement, vous pouvez vous aider des journaux et des traces, des données de contrôle, des données concernant le matériel et des notes sur l'édition.

Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance
Pour isoler et résoudre les problèmes rencontrés avec vos produits IBM, vous pouvez utiliser les informations relatives à l'identification et la résolution des incidents. Ces informations contiennent des consignes concernant l'utilisation des ressources de détermination des problèmes fournies avec vos produits IBM, y compris WebSphere DataPower XC10 Appliance.

Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance
Les fichiers journaux associés à WebSphere DataPower XC10 Appliance sont stockés sur le dispositif. Les journaux consultables peuvent être consultés directement à partir du dispositif avec l'interface utilisateur ou téléchargés sur votre système de fichiers local pour examen.

Téléchargement de données d'audit
L'activité d'audit est consignée par le système WebSphere DataPower XC10 Appliance. L'activité des utilisateurs concernant un ensemble d'objets sujets à audit est mémorisée de sorte à garantir une couverture adéquate de l'audit.

Analyse des données de journal et de trace
Vous pouvez utiliser les outils d'analyse de journal pour analyser le fonctionnement de l'environnement d'exécution et résoudre les problèmes qui y apparaissent.

Surveillance de la température du matériel
Des capteurs mesurent en permanence la température de divers composants internes du dispositif. Ces températures peuvent être surveillées facilement à partir d'un seul panneau à l'aide de l'interface utilisateur.

Surveillance du statut des interfaces Ethernet à l'aide d'une connexion série
Si votre système IBM® WebSphere DataPower XC10 Appliance rencontre des incidents au niveau des réseaux, vous pouvez afficher des informations sur l'activité et le statut des interfaces Ethernet. Cette rubrique décrit comment afficher ces informations à l'aide d'une connexion série établie avec le dispositif. Notez que vous pouvez aussi afficher ces informations au moyen de l'interface utilisateur.

Vérification des connexions sortantes du dispositif
A l'aide de la fonction Connexion sortante, vous pouvez vérifier si une adresse réseau est accessible à partir du dispositif.

Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Vous pouvez exécuter des commandes pour redémarrer le matériel du dispositif, rétablir les paramètres usine du dispositif ou arrêter le dispositif.

Traitement des problèmes d'interblocage
Cette rubrique décrit des scénarios d'interblocage courants et des solutions pour les éviter.

2.5+ **Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition**
Le scénario décrit est un exemple de transaction multipartition qui est à l'origine d'une exception de dépassement de délai de verrouillage. Selon l'état de la transaction, les solutions montrent la façon dont vous pouvez manuellement résoudre ce problème.

Traitement des problèmes d'intégration du cache
Utilisez ces informations pour traiter les problèmes de la configuration de l'intégration du cache, y compris ceux associés aux configurations de session HTTP et de cache dynamique.

Traitement des problèmes d'installation
Utilisez ces informations pour traiter les problèmes liés à l'installation et aux mises à jour.

Traitement des problèmes d'administration
Utilisez les informations suivantes pour traiter les problèmes d'administration, notamment le démarrage et l'arrêt des serveurs, en utilisant l'utilitaire **xscmd**, etc.

Notes sur l'édition
Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Pour isoler et résoudre les problèmes rencontrés avec vos produits IBM, vous pouvez utiliser les informations relatives à l'identification et la résolution des incidents. Ces informations contiennent des consignes concernant l'utilisation des ressources de détermination des problèmes fournies avec vos produits IBM, y compris WebSphere DataPower XC10 Appliance.

Techniques d'identification et de résolution d'incidents

L'*identification et la résolution des incidents* est une approche systématique de la phase de résolution d'un problème. L'objectif est de déterminer les raisons pour lesquelles une opération ne fonctionne pas comme prévu et d'expliquer la procédure à suivre pour résoudre le problème. Certaines techniques courantes peuvent être utiles pour la tâche d'identification d'incident.

Recherche dans les bases de connaissances

Vous pouvez souvent trouver des solutions à vos problèmes en effectuant des recherches dans les bases de connaissances IBM. Vous pouvez optimiser vos résultats en utilisant les ressources disponibles, les outils de support et les méthodes de recherche.

Obtention de correctifs

Il existe peut-être un correctif produit permettant de résoudre votre problème.

Comment prendre contact avec le service de support IBM

Le service de support IBM assure l'assistance relative aux problèmes rencontrés avec les produits, répond aux questions (FAQ) et aide les utilisateurs à résoudre les incidents liés au produit.

Collecte d'informations de diagnostic pour le support IBM

Vous pouvez utiliser l'interface de ligne de commande pour collecter des journaux à envoyer au support IBM.

Echange d'informations avec IBM

Pour diagnostiquer ou identifier un incident, vous avez peut-être besoin de fournir au service de support IBM des données et des informations issues de votre système. Dans d'autres cas, le service de support IBM peut vous fournir des outils ou utilitaires à utiliser pour l'identification de l'incident.

Abonnement aux mises à jour de support

Pour rester informé des informations importantes sur les produits IBM que vous utilisez, vous pouvez vous abonner aux mises à jour.

Rubrique parent : Traitement des incidents

Techniques d'identification et de résolution d'incidents

L'*identification et la résolution des incidents* est une approche systématique de la phase de résolution d'un problème. L'objectif est de déterminer les raisons pour lesquelles une opération ne fonctionne pas comme prévu et d'expliquer la procédure à suivre pour résoudre le problème. Certaines techniques courantes peuvent être utiles pour la tâche d'identification d'incident.

La première étape du processus consiste à décrire complètement le problème. La description de l'incident vous permet, et permet également au technicien de maintenance IBM, de savoir où commencer à chercher la cause de l'incident. A ce stade, vous devez vous poser les questions de base suivantes :

- Quels sont les symptômes du problème ?
- Où survient le problème ?
- Quand survient le problème ?
- Dans quelles conditions le problème survient-il ?
- Le problème peut-il être reproduit ?

Les réponses à ces questions permettent généralement de décrire avec précision le problème, ce qui vous permet de le résoudre.

Quels sont les symptômes du problème ?

Lorsque vous commencez à décrire un problème, la question la plus évidente est "Quel est le problème ?" Cette question peut sembler simple ; toutefois, vous pouvez la scinder en plusieurs questions plus concentrées qui permettent d'avoir une vue plus descriptive du problème. Ces questions sont notamment les suivantes :

- Qui ou qu'est-ce qui a généré un rapport de problème ?
- Quels sont les codes et les messages d'erreur ?
- Comment le système a échoué ? Par exemple, boucle, arrêt, plantage, dégradation des performances, résultat incorrect.

Où survient le problème ?

Il n'est pas toujours aisé de déterminer où se trouve le problème, mais il s'agit pourtant de l'une des étapes les plus importantes. De nombreuses couches technologiques peuvent exister entre le composant qui signale l'incident et le composant défaillant. Les réseaux, la grille de données et les serveurs ne sont que quelques exemples de composants à prendre en compte lorsque vous cherchez à en savoir plus sur les problèmes rencontrés.

Les questions suivantes vous aident à vous concentrer sur l'endroit où survient le problème pour isoler la couche de problème :

- Le problème est-il spécifique à une plateforme ou à un système d'exploitation, ou concerne-t-il plusieurs plateformes ou systèmes d'exploitation ?
- L'environnement et la configuration actuels sont-ils pris en charge ?
- Tous les utilisateurs rencontrent-ils ce problème ?
- (Pour les installations multisite.) Tous les sites rencontrent-ils ce problème ?

Ce n'est pas parce qu'une couche signale le problème que celui-ci provient obligatoirement de cette couche. Pour identifier l'endroit d'où provient un problème, vous devez comprendre l'environnement dans lequel ce problème se produit. Prenez quelques instants pour décrire complètement l'environnement du problème, notamment le système d'exploitation et sa version, tous les logiciels correspondants et leur version, ainsi que les informations sur le matériel. Vérifiez que la configuration de votre environnement est prise en charge ; de nombreux problèmes peuvent être provoqués par l'utilisation de logiciels incompatibles qui ne sont pas destinés à être exécutés ensemble ou dont l'exécution simultanée n'a pas été testée.

Quand survient le problème ?

Dressez la liste chronologique des événements qui ont conduit à l'apparition de l'incident, en particulier si l'incident ne s'est produit qu'une seule fois. Vous pouvez très aisément dresser une liste chronologique en procédant à l'envers : partez du moment où une erreur a été signalée (aussi précisément que possible, même à la milliseconde près), et remontez en arrière à l'aide des journaux et des informations disponibles. Il vous suffit généralement de remonter jusqu'au premier événement suspicieux signalé dans un journal de diagnostic.

Pour dresser une liste chronologique détaillée des événements, répondez aux questions suivantes :

- Le problème se produit-il uniquement à certains moments du jour ou de la nuit ?
- Selon quelle fréquence le problème se produit-il ?
- Quelle séquence d'événements a provoqué la survenue du problème ?

- Le problème s'est-il produit après un changement apporté à l'environnement, tel qu'une mise à niveau ou l'installation d'un logiciel ou d'un matériel ?

En répondant à ces questions, vous définissez un cadre de référence dans lequel mener vos recherches.

Dans quelles conditions le problème survient-il ?

Il est très important de savoir quelles applications et quels systèmes étaient en cours d'exécution lorsque l'incident s'est produit. Les questions suivantes concernant l'environnement vous aideront à identifier la cause de l'incident :

- Le problème se produit-il toujours lorsque vous effectuez la même tâche ?
- Est-ce qu'une séquence d'événements doit se produire pour provoquer le problème ?
- Est-ce que l'exécution d'autres applications échoue également ?

En répondant à ces questions, vous pouvez décrire l'environnement dans lequel l'incident se produit, et identifier les éventuelles dépendances. Notez cependant que si plusieurs incidents se produisent de manière quasi-simultanée, cela ne signifie pas nécessairement que ces incidents sont liés.

Le problème peut-il être reproduit ?

Du point de vue de l'identification et de la résolution, le problème idéal est celui qui peut être reproduit. En général, lorsqu'un problème peut être reproduit, vous disposez d'un plus grand nombre d'outils ou de procédures pour en savoir plus sur ces problèmes. Par conséquent, les problèmes que vous pouvez reproduire sont souvent plus faciles à déboguer et résoudre.

Toutefois, ces problèmes présentent un inconvénient : si le problème en question a un impact commercial considérable, vous ne voulez pas qu'il se reproduise. Si possible, recréez l'incident dans un environnement de test ou de développement qui vous offre généralement plus de souplesse et de contrôle.

- Le problème peut-il être recréé sur un système de test ?
- Est-ce que plusieurs utilisateurs ou applications rencontrent le même type de problème ?
- Le problème peut-il être recréé en exécutant une seule commande, un jeu de commandes ou une application particulière ?

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Recherche dans les bases de connaissances

Vous pouvez souvent trouver des solutions à vos problèmes en effectuant des recherches dans les bases de connaissances IBM. Vous pouvez optimiser vos résultats en utilisant les ressources disponibles, les outils de support et les méthodes de recherche.

Pourquoi et quand exécuter cette tâche

Vous pouvez trouver des informations utiles en effectuant des recherches dans le centre de documentation WebSphere DataPower XC10 Appliance. Cependant, il est parfois nécessaire de pousser les recherches plus loin pour trouver des réponses à vos questions ou résoudre certains problèmes.

Procédure

Pour rechercher les informations dont vous avez besoin dans les bases de connaissances, utilisez une ou plusieurs des approches suivantes :

- Recherchez un contenu à l'aide d'IBM® Support Assistant (ISA).

  ISA est un plan de travail de serviçabilité gratuit qui vous permet de répondre à certaines questions et de résoudre les problèmes relatifs aux logiciels IBM. Vous trouverez les instructions de téléchargement et d'installation d'ISA sur le site Web d'ISA.

- Recherchez le contenu dont vous avez besoin à l'aide du portail de support IBM.

  Le portail de support IBM est une vue centralisée et unifiée de tous les outils de support technique et de toutes les informations relatives à l'ensemble des systèmes, logiciels et services IBM. Le portail de support IBM permet d'accéder au portefeuille d'assistance électronique d'IBM à partir d'un emplacement unique. >Vous pouvez en personnaliser les pages pour cibler les informations et les ressources dont vous avez besoin pour empêcher un problème ou en résoudre plus rapidement un. Familiarisez-vous avec le portail de support IBM en visionnant les vidéos de démonstration (https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos) sur cet outil. Ces vidéos vous présentent le portail de support IBM, explorent la fonction d'identification et de résolution des problèmes et d'autres ressources et expliquent comment personnaliser la page en déplaçant, ajoutant et supprimant des portlets.

- Recherchez du contenu relatif à WebSphere DataPower XC10 Appliance en utilisant l'une des ressources techniques supplémentaires suivantes :
  - WebSphere DataPower XC10 Appliance - Notes sur l'édition
  - Site Web d'assistance WebSphere DataPower XC10 Appliance
  - Forum WebSphere DataPower XC10 Appliance
- Recherchez un contenu à l'aide de la fonction de recherche générique IBM. Vous pouvez utiliser la fonction de recherche générique IBM en saisissant votre chaîne de recherche dans la zone de recherche, dans la partie supérieure de toute page ibm.com.
- Recherchez un contenu à l'aide de tout moteur de recherche externe, tel que Google, Yahoo ou Bing. Si vous utilisez un moteur de recherche externe, vos résultats risquent d'inclure davantage d'informations en dehors du domaine ibm.com. Cependant, en consultant certains forums et blogues en dehors du domaine ibm.com, vous pouvez parfois y trouver des informations utiles pour résoudre des problèmes sur les produits IBM.

  Conseil : Incluez "IBM" et le nom du produit dans votre recherche si vous recherchez des informations sur un produit IBM.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Obtention de correctifs

Il existe peut-être un correctif produit permettant de résoudre votre problème.

Procédure

Pour rechercher et installer des correctifs, procédez comme suit :

1. Procurez-vous les outils requis pour obtenir le correctif. Utilisez IBM Update Installer pour installer et appliquer plusieurs types de packages de maintenance pour WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client. Ce programme d'installation de mises à jour étant soumis à des opérations de maintenance régulières, veuillez utiliser sa dernière version.
2. Déterminez le correctif dont vous avez besoin. Pour sélectionner le correctif le plus récent, voir Recommended fixes for WebSphere DataPower XC10 Appliance. Lors de la sélection d'un correctif, le document de téléchargement pour ce correctif s'ouvre.
3. Téléchargez le correctif. Dans le document de téléchargement, cliquez sur le lien relatif au correctif le plus récent dans la section "Download package".
4. Appliquez le correctif. Suivez les instructions de la section "Installation Instructions" du document de téléchargement.
5. Abonnez-vous pour recevoir des notifications hebdomadaires par courrier électronique sur les correctifs et d'autres informations du service de support IBM.

   Obtention de correctifs à partir de Fix Central
   Vous pouvez utiliser Fix Central pour rechercher les correctifs recommandés par le service de support IBM pour divers produits, y compris WebSphere DataPower XC10 Appliance. Avec Fix Central, vous pouvez rechercher, sélectionner, commander et télécharger des correctifs pour votre système et avez le choix entre plusieurs options de distribution. Un correctif de produit WebSphere DataPower XC10 Appliance peut être disponible pour résoudre votre incident.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Obtention de correctifs à partir de Fix Central

Vous pouvez utiliser Fix Central pour rechercher les correctifs recommandés par le service de support IBM pour divers produits, y compris WebSphere DataPower XC10 Appliance. Avec Fix Central, vous pouvez rechercher, sélectionner, commander et télécharger des correctifs pour votre système et avez le choix entre plusieurs options de distribution. Un correctif de produit WebSphere DataPower XC10 Appliance peut être disponible pour résoudre votre incident.

Procédure

Pour rechercher et installer des correctifs, procédez comme suit :

1. Procurez-vous les outils requis pour obtenir le correctif. Si votre programme d'installation de mises à jour n'est pas installé, procurez-le vous. Vous pouvez télécharger le programme d'installation à partir de Fix Central. Ce site fournit des instructions de téléchargement, d'installation et de configuration pour le programme d'installation de mises à jour.
2. Sélectionnez le produit, puis cochez une ou plusieurs cases correspondant à l'incident que vous souhaitez résoudre.
3. Identifiez et sélectionnez le correctif requis.
4. Téléchargez le correctif.
    a. Ouvrez le document de téléchargement, puis suivez le lien indiqué dans la section "Download Package".
    b. Lorsque vous téléchargez le fichier, vérifiez que le nom du fichier de maintenance n'est pas modifié. Cette modification peut être intentionnelle ou découler de certains navigateurs Web ou utilitaires de téléchargement.
5. Appliquez le correctif.
    a. Suivez les instructions de la section "Installation Instructions" du document de téléchargement.
    b. Pour plus d'informations, reportez-vous à la rubrique "Installing fixes with the Update Installer" dans la documentation du produit.
6. Facultatif : Abonnez-vous pour recevoir des notifications hebdomadaires par courrier électronique sur les correctifs et d'autres mises à jour du service de support IBM.

Rubrique parent : Obtention de correctifs

Comment prendre contact avec le service de support IBM

Le service de support IBM assure l'assistance relative aux problèmes rencontrés avec les produits, répond aux questions (FAQ) et aide les utilisateurs à résoudre les incidents liés au produit.

Avant de commencer

Une fois que vous avez essayé de rechercher votre réponse ou votre solution à l'aide d'autres options d'auto-assistance, telles que les notes sur l'édition, vous pouvez contacter le service de support IBM. Pour contacter le service de support IBM, votre société ou organisation doit avoir souscrit en contrat de maintenance IBM en cours de validité, et vous devez être autorisé à soumettre des problèmes à IBM. Pour obtenir des informations sur les types de support disponibles, reportez-vous à la rubrique Support portfolio du "*Software Support Handbook*".

Procédure

Pour signaler un problème au service de support IBM, procédez comme suit :

1. Définissez la nature du problème, collectez des informations générales, puis identifiez le niveau de gravité du problème. Pour plus d'informations, reportez-vous à la rubrique Getting IBM support du *Software Support Handbook*.
2. Collectez des informations de diagnostic.
3. Soumettez le problème au service de support IBM de l'une des manières suivantes :
    - Avec IBM® Support Assistant (ISA). .
    - En ligne via le Portail de support IBM : Vous pouvez ouvrir, mettre à jour et afficher toutes vos demandes de service à partir du portlet Demande de service de la page Demande de service.
    - Par téléphone : Pour connaître le numéro à appeler dans votre pays ou région, reportez-vous à la page Web Directory of worldwide contacts.

Résultats

Si le problème signalé concerne un incident logiciel, ou de la documentation manquante ou incorrecte, le service de support IBM crée un rapport officiel d'analyse de programme (APAR). Ce rapport expose le problème en détail. Dans la mesure du possible, le service de support IBM propose une solution palliative que vous pouvez mettre en oeuvre en attendant la finalisation de l'APAR et la mise à disposition d'un correctif . IBM publie chaque jour sur le site Web du service de support IBM, les APAR traités afin que les utilisateurs susceptibles de rencontrer le même problème puissent prendre connaissance de la solution à appliquer.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Collecte d'informations de diagnostic pour le support IBM

Vous pouvez utiliser l'interface de ligne de commande pour collecter des journaux à envoyer au support IBM.

Avant de commencer

Vous devez disposer d'une connexion d'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.

Procédure

- Collectez uniquement les informations d'état et de configuration du dispositif. Dans l'interface de ligne de commande, exécutez la commande suivante :

  ```
  platform collect-pd <PDFileName>
  ```

  Si vous n'indiquez pas un nom de fichier, le fichier `collect-pd.txt` est créé par défaut. Ce fichier contient la sortie des commandes de statut de dispositif et des détails concernant la configuration réseau.

- Collectez le journal et les fichiers de trace du dispositif. Dans l'interface de ligne de commande, exécutez la commande suivante :

  ```
  platform must-gather  <tarfilename> [<PDFileName>]
  ```

  La commande crée un fichier tar qui inclut le journal et les fichiers de trace du dispositif. Vous devez spécifier un nom de fichier tar pour exécuter la commande. Cette commande exécute la commande **platform collect-pd** avant de créer le fichier tar.

- Effacez les fichiers journaux. Si votre dispositif est en cours d'exécution depuis une longue période, les fichiers journaux peuvent être volumineux. Si le fichier tar généré par la commande **platform must-gather** est d'une taille supérieure à un gigaoctet, vous pouvez exécuter la commande suivante pour remettre à zéro l'ensemble des fichiers journaux :

  ```
  clear-logs
  ```

- Copiez les fichiers générés par le dispositif pour les envoyer au support IBM.

  1. Affichez la liste des fichiers que vous avez généré à l'aide de la commande suivante :

     ```
     Console> file list

     logsTest.tgz 1810748334 bytes created 2011-06-08 08:22:17-0500

     collect-pd.txt 17477 bytes created 2011-06-08 08:18:34-0500
     ```

  2. Copiez les fichiers journaux du dispositif avec la commande suivante :

     ```
     Console> file put logsTest.tgz
     scp://root@linux010.myco.com:/opt/cp/logsTest.tgz

     Password:*******
     ```

Que faire ensuite

Envoyez les fichiers au support IBM. Pour plus d'informations, voir Echange d'informations avec IBM.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Echange d'informations avec IBM

Pour diagnostiquer ou identifier un incident, vous avez peut-être besoin de fournir au service de support IBM des données et des informations issues de votre système. Dans d'autres cas, le service de support IBM peut vous fournir des outils ou utilitaires à utiliser pour l'identification de l'incident.
Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Envoi d'informations au service de support IBM

Pour réduire le temps requis pour la résolution de votre incident, vous pouvez envoyer des informations de trace et de diagnostic au service de support IBM.

Procédure

Pour soumettre des informations de diagnostic au service de support IBM, procédez comme suit :

1. Ouvrez un enregistrement PMR.
2. Collectez les données de diagnostic dont vous avez besoin. Les données de diagnostic vous aident à réduire le temps de résolution de votre enregistrement PMR. Vous pouvez collecter les données de diagnostic manuellement ou automatiquement :
   - Collecte manuelle des données.
   - Collecte automatique des données.
3. Compressez les fichiers à l'aide du format de fichier `.zip` ou `.tar`.
4. Transférez les fichiers à IBM. Vous pouvez utiliser l'une des méthodes suivantes pour transférer les fichiers à IBM :
   - IBM® Support Assistant
   - L'outil de demande de service
   - Méthodes de téléchargement de données standard : FTP, HTTP
   - Méthodes de téléchargement de données sécurisées : FTPS, SFTP, HTTPS
   - Courrier électronique

   Si vous utilisez un produit z/OS et que vous faites appel à ServiceLink / IBMLink pour soumettre des enregistrements PMR, vous pouvez envoyer les données de diagnostic au service de support IBM dans un courrier électronique ou via FTP.

   Toutes ces méthodes d'échange de données sont expliquées sur le site Web du service de support IBM.

Réception d'informations du service de support IBM

Il arrive parfois qu'un technicien de maintenance IBM vous demande de télécharger des outils de diagnostic ou d'autres fichiers. Vous pouvez utiliser FTP pour télécharger ces fichiers.

Avant de commencer

Assurez-vous que votre technicien de maintenance IBM vous a fourni le serveur préféré à utiliser pour le téléchargement des fichiers, ainsi que les noms de répertoire et de fichier exacts auxquels accéder.

Procédure

Pour télécharger des fichiers à partir du service de support IBM, procédez comme suit :

1. Utilisez FTP pour vous connecter au site indiqué par votre technicien de maintenance IBM, puis connectez-vous en tant qu'utilisateur `anonymous`. Utilisez votre adresse électronique comme mot de passe :
2. Accédez au répertoire approprié :
   a. Accédez au répertoire `/fromibm`.

   ```
   cd fromibm
   ```

   b. Accédez au répertoire fourni par votre technicien de maintenance IBM.

   ```
   cd nomdurépertoire
   ```

3. Activez le mode binaire pour votre session.

   ```
   binary
   ```

4. Utilisez la commande **get** pour télécharger le fichier indiqué par votre technicien de maintenance IBM.

```
get nomfichier.extension
```

5. Mettez fin à votre session FTP.

```
quit
```

Abonnement aux mises à jour de support

Pour rester informé des informations importantes sur les produits IBM que vous utilisez, vous pouvez vous abonner aux mises à jour.

Pourquoi et quand exécuter cette tâche

En vous abonnant pour recevoir les mises à jour sur le produit, vous pouvez recevoir les mises à jour et informations techniques importantes sur des ressources et outils de support IBM spécifiques. Vous pouvez vous abonner aux mises à jour à l'aide de l'une des deux approches suivantes :

Abonnements aux média sociaux

Le flux RSS suivant est disponible pour le produit :

- Flux RSS pour le forum WebSphere DataPower XC10 Appliance

Pour des informations générales sur RSS, dont la procédure d'initiation et la liste des pages Web IBM compatibles RSS, visitez le site Flux RSS du service de support logicielIBM.

Mes notifications

Avec Mes notifications, vous pouvez vous abonner aux mises à jour de support de tout produit IBM. Mes notifications remplace Mon Support, qui est un outil similaire que vous avez peut-être utilisé dans le passé. Avec Mes Notifications, vous pouvez indiquer que vous souhaitez recevoir des annonces quotidiennes ou hebdomadaires par courrier électronique. Vous pouvez spécifier le type d'informations à recevoir (par exemple, des publications, des astuces et des conseils, des notifications flash sur les produits (ou alertes), des téléchargements et des pilotes). Mes notifications vous permet de personnaliser et classer les produits sur lesquels vous souhaitez être informé et les méthodes de distribution qui répondent le mieux à vos besoins.

Procédure

Pour vous abonner aux mises à jour de support :

1. Abonnez-vous au flux RSS pour le forum WebSphere DataPower XC10 Appliance.
    a. Dans la page d'abonnement, cliquez sur l'icône du flux RSS.
    b. Sélection l'option que vous souhaitez utiliser pour vous abonner au flux.
    c. Cliquez sur **S'abonner**.
2. Abonnez-vous à Mes notifications en accédant au Portail de support IBM® et cliquez sur **Mes notifications** dans le portlet **Notifications**.
3. Ouvrez une session à l'aide de votre ID IBM et de votre mot de passe, puis cliquez sur **Soumettre**.
4. Indiquez les types de mise à jour à recevoir et la manière de les recevoir.
    a. Cliquez sur l'onglet **S'abonner**.
    b. Sélectionnez la marque de logiciel ou le type de matériel approprié.
    c. Sélectionnez un ou plusieurs produits par nom, puis cliquez sur **Continuer**.
    d. Sélectionnez vos préférences de mode de réception des mises à jour (e-mail, en ligne dans un dossier désigné ou comme flux RSS ou Atom).
    e. Sélectionnez les types de mise à jour de documentation à recevoir, par exemple, les nouvelles informations sur les téléchargements de produit et les commentaires des groupes de discussion.
    f. Cliquez sur **Soumettre**.

Résultats

Tant que vous ne modifiez pas vos flux RSS et vos préférences Mes notifications, vous recevez les notifications des mises à jour que vous avez demandées. Vous pouvez modifier vos préférences si nécessaire (par exemple, si vous arrêtez d'utiliser un produit et commencez à en utiliser un autre).

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Informations connexes

- Flux RSS du service de support logiciel IBM
- S'abonner aux mises à jour de contenu du support Mes notifications
- Mes notifications pour le service de support logiciel IBM
- Présentation de Mes notifications pour le service de support logiciel IBM

Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance

Les fichiers journaux associés à WebSphere DataPower XC10 Appliance sont stockés sur le dispositif. Les journaux consultables peuvent être consultés directement à partir du dispositif avec l'interface utilisateur ou téléchargés sur votre système de fichiers local pour examen.

Avant de commencer

Vous devez disposer des droits d'accès d'administrateur du dispositif pour réaliser ces opérations.

Pourquoi et quand exécuter cette tâche

Les données de journal sont consignées directement sur le dispositif. Vous pouvez envoyer le fichier `trace.zip` à l'équipe d'assistance IBM®.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Dispositif** > **Traitement des problèmes** et développez **Consignation au journal**. Les données des journaux sont stockées sur le dispositif. En développant la section **Consignation au journal**, vous pouvez accéder aux journaux disponibles à l'aide de l'afficheur de journal et vous pouvez les télécharger sur votre système de fichier pour les analyser de façon plus approfondie.
2. Visualisez les fichiers journaux en cours. Pour afficher les journaux, cliquez sur l'un des liens suivants :
   - **Afficher le fichier d'erreurs actuel**
   - **Afficher le fichier de trace actuel**
   - 2.5+ **Afficher le fichier de santé actuel**

   Une nouvelle fenêtre de navigateur s'ouvre pour l'afficheur de journal. Cet afficheur présente les 10 dernières lignes du journal sélectionné. Les nouvelles entrées au journal s'y ajoutent au fur et à mesure qu'elles sont consignées. Plusieurs actions permettent de contrôler le comportement de l'afficheur de journal.

   a. Cliquez sur **Pause** pour arrêter l'affichage de nouvelles entrées du journal. Cette action n'est disponible que si l'afficheur acceptait de nouvelles entrées.

   b. Cliquez sur **Redémarrer** pour permettre l'ajout de nouvelles entrées. Cette action n'est disponible que si l'afficheur n'acceptait plus de nouvelles entrées.

   c. Cliquez sur **Effacer** pour effacer toutes les données de l'afficheur de journal. Cette action est disponible que l'afficheur de journal accepte ou non de nouvelles entrées.
3. Cliquez sur **Télécharger les fichiers journaux** pour enregistrer tous les journaux disponibles sur votre système de fichiers. Si vous devez visualiser les informations concernant les événements qui se sont produits, vous devez utiliser ce lien. Une fenêtre s'affiche pour vous permettre d'ouvrir le fichier compressé ou de l'enregistrer sur votre système de fichiers. Le fichier`trace.zip` contient tous les fichiers journaux à fournir à l'équipe de support.

   Un ensemble de fichiers CSV est inclus dans le fichier `trace.zip` qui vous permet de suivre les données historiques des serveurs sur le dispositif. Dans le fichier `trace.zip`, les fichiers CSV se trouvent dans le répertoire *nom_serveur*`/logs`. Les fichiers sont intitulés `jvmstats.log`, `mapstats.log` et `ogstats.log`. Vous pouvez collecter les informations suivantes à partir des fichiers CSV : utilisation de la mémoire par la mappe ou le serveur, taux de réussite de la mappe, temps de transaction, débit.
4. Dans la section **Configurer les niveaux de trace**, vous pouvez afficher ou modifier les niveaux de trace. Vous pouvez modifier les niveaux de trace de la **console d'administration** ou de la **Grille de données**. Pour la console d'administration, vous pouvez modifier la sortie du gestionnaire de journalisation par défaut en la fixant aux niveaux de trace suivants :
   - OFF
   - SEVERE
   - WARNING
   - INFO
   - FINE

   a. Ajout d'une chaîne de trace. Cliquez sur **Ajouter un paramètre de trace** et ajoutez une chaîne de trace valide. Le niveau de trace pour une nouvelle chaîne de trace est défini par défaut à *INFO*.

   b. Suppression d'une chaîne de trace. Cliquez sur l'icône de suppression ( ) en regard d'une chaîne de trace pour supprimer cette dernière.

   c. Modification d'une chaîne de trace. Cliquez sur le niveau de trace et sélectionnez un nouveau niveau de trace. Cliquez sur **Sauvegarder** pour valider le nouveau niveau de trace pour la

chaîne spécifiée.

Que faire ensuite

Si la console Web n'est pas disponible, vous pouvez toujours récupérer le fichier `trace.zip`. Procédez comme suit à partir de la ligne de commande pour extraire les fichiers journaux à fournir au support IBM :

1. Collectez les fichiers journaux.

   ```
   Console> platform must-gather
   ```

2. Affichez la liste des fichiers journaux générés.

   ```
   Console> file list
   ```

3. Copiez les journaux du dispositif.

   ```
   Console> file put <nom-de-mon-choix>.tgz
   scp://root@remoteMachine.myco.com:/myRemoteMachinePath/<nom-de-mon-choix>.tgz
   ```

Envoi d'enregistrements de journal WebSphere DataPower XC10 Appliance à un système UNIX distant à l'aide de syslog
Vous pouvez activer la consignation à distance pour envoyer des entrées de journal sur un serveur distant. La consignation à distance peut être utile pour sauvegarder des informations à des fins d'audit.

Rubrique parent : Traitement des incidents

Tâches associées:
Surveillance à l'aide de fichiers CSV

Référence associée:
Définition des statistiques des fichiers CSV

Envoi d'enregistrements de journal WebSphere DataPower XC10 Appliance à un système UNIX distant à l'aide de syslog

Vous pouvez activer la consignation à distance pour envoyer des entrées de journal sur un serveur distant. La consignation à distance peut être utile pour sauvegarder des informations à des fins d'audit.

Avant de commencer

- Vous devez disposer des droits d'administration de dispositif.
- Consultez la documentation de votre système d'exploitation pour obtenir des instructions spécifiques concernant la configuration du serveur daemon syslog et la réception d'enregistrements syslog.
- WebSphere DataPower XC10 Appliance prend en charge le protocole syslog RFC 3164.
- Les enregistrements de journal sont envoyés à l'aide du protocole de datagramme utilisateur (UDP) et sont par conséquent de type "fire-and-forget", ce qui signifie que la livraison des messages n'est pas garantie (il s'agit là d'une limitation du protocole UDP).
- Notez les processus serveur de conteneur qui sont susceptibles de faire l'objet d'une consignation et d'un envoi sur le système distant sur WebSphere DataPower XC10 Appliance:
  - **`cs`**
  - **`xsServer00-xsServerXY`** : où **`XY`** représente le nombre de processus de conteneur sur le dispositif.
  - **`xsa.admin`**

  Lorsque la consignation à distance est activée, les processus peuvent être envoyés à un syslogd. Lorsqu'elle est désactivée, aucun de ces processus n'est envoyée à un syslogd. Si l'un des processus est arrêté ou temporairement indisponible, la journalisation se poursuit pour les processus qui sont encore actifs. Si vous soupçonnez des problèmes au niveau de syslog dans WebSphere DataPower XC10 Appliance qui doivent être résolus, vous pouvez ajouter un niveau de traçage **`SYSLOG=all`** via l'interface utilisateur du dispositif.

Pourquoi et quand exécuter cette tâche

Utilisez la consignation à distance pour l'archivage à long terme des événements consignés archivés. WebSphere DataPower XC10 Appliance conserve les enregistrements de journal dans un nombre restreint de fichiers journaux, pour une durée limitée afin d'économiser de l'espace. Configurez et activez la consignation à distance dans WebSphere DataPower XC10 Appliance si vous souhaitez archiver les enregistrements de journal pendant plus de temps ou si vous souhaitez analyser des enregistrements de journal archivés au-delà de la date d'expiration des journaux.

Procédure

1. Dans l'interface utilisateur du dispositif, cliquez sur **Dispositif** > **Traitement des problèmes** > **Consignation au journal**.
2. Dans la section **Configurer la consignation à distance**, sélectionner **Activer la consignation à distance**. Cela permet d'activer la consignation à distance aux fins d'analyse des données d'historique. Vous devez disposer d'un serveur syslogd qui écoute les événements et les capture. Définissez les paramètres suivants :
   a. **Hôte distant** - Indique le nom d'hôte ou l'adresse IP du serveur syslogd distant sur lequel vous voulez envoyer les enregistrements de journal. La valeur ne peut pas être "localhost", "locahost-v6", "127.0.0.1" ou "::1". Assurez-vous que vous pouvez envoyer une commande ping à l'hôte ou l'adresse IP du dispositif.
   b. **Port distant** - Indique le numéro de port du serveur syslogd sur lequel vous voulez envoyer les enregistrements de journal. Les valeurs valides sont comprises entre `0 et 65535`, et la valeur par défaut est 512.
   c. **Seuil** - Indique le seuil de gravité des messages à envoyer au serveur de journalisation distant. Pour envoyer les messages d'avertissement et d'erreur grave, entrez la valeur **WARNING**. Pour n'envoyer que les messages d'erreur grave, sélectionnez **SEVERE**.
   d. **Fonction syslog** - Indique la fonction de journalisation syslog qui est utilisée pour l'envoi des message de journal. Cette valeur détermine le fichier dans lequel le message de journal sera placé par le serveur démon syslog qui s'exécute sur le système distant. Par exemple, si vous définissez la fonction sur **user**, la plupart des serveurs démons syslog stockeront le message dans un fichier appelé `/var/log/user.log`. Si vous sélectionnez **mail**, le serveur démon syslog stockera le message dans le fichier `/var/log/mail.log`. La destination de vos enregistrements de journal dépend de la manière dont vous avez configuré le serveur démon syslog. Pour plus d'informations concernant la configuration de syslogd, reportez-vous aux instructions pour votre système d'exploitation.

   Pour modifier les paramètres, cliquez sur **Appliquer les modifications**. Les modifications sont appliquées à tous les processus concernés par les messages syslog.

Résultats

Les enregistrements de journal du dispositif sont envoyés au serveur de consignation à distance configuré pour l'archivage et l'analyse.

Rubrique parent : Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance

Téléchargement de données d'audit

L'activité d'audit est consignée par le système WebSphere DataPower XC10 Appliance. L'activité des utilisateurs concernant un ensemble d'objets sujets à audit est mémorisée de sorte à garantir une couverture adéquate de l'audit.

Pourquoi et quand exécuter cette tâche

Les données d'audit qui consignent l'activité des utilisateurs concernant des objets spécifiques sujets à audit sont stockées sur le dispositif. Reportez-vous au tableau suivant pour information sur les objets concernés inclus dans les données d'audit.

Tableau 1. Objets sujets à audit

Objets	Ajout/Création	Supprimer	Mise à jour	Autre
Intégration	Y	Y	N/D	
Groupe	Y	Y	Y	
session	N/D	N/D	N/D	Expiration, connexion
Utilisateur	Y	Y	Y	
Zone	Y	Y	Y	
Activation de zone	Y	Y	Y	

L'intégrité des données est d'une importance cruciale pour des procédures d'audit adéquates. Pour garantir l'intégrité des données, les données d'audit doivent être extraites du dispositif au moment où elles sont requises. Après avoir téléchargé les données d'audit, les méthodes utilisées pour garantir leur intégrité relèvent de votre responsabilité. Procédez comme suit pour extraire vos données d'audit du dispositif.

Procédure

1. Ouvrez la section **Audit** de l'interface utilisateur. Cliquez sur**Dispositif** > **Identification et résolution des incidents**. Développez la section **Audit**. Vous pouvez consulter les données d'audit disponibles pour votre dispositif en développant la section **Audit**.
2. Téléchargez les données d'audit. Vous pouvez télécharger l'intégralité des données ou une version filtrée.
   - **Télécharger toutes les données**

     Cliquez sur **Télécharger toutes les données** pour télécharger le fichier `audit.zip` . Le fichier `audit.zip` contient toutes les données actuellement disponibles sur le dispositif. Si vous souhaitez consulter les données d'audit au format CSV ou uniquement une partie des données disponibles, vous pouvez définir en conséquence la plage de données et télécharger les données filtrées.

   - **Télécharger les données filtrées**

     Ajustez la plage de dates et définissez les paramètres de fuseau horaire. La plage de dates et les paramètres de fuseau horaire sont utilisés pour générer les données filtrées. Ces paramètres ne s'appliquent qu'aux données filtrées et non pas lorsque vous téléchargez la totalité des données. La valeur de fuseau horaire par défaut est celle associée au dispositif. La plage de dates par défaut couvre les données des derniers 30 jours. En ajustant le fuseau horaire et la plage de dates, vous déterminez les données d'audit qui seront incluses.

     Cliquez sur **Télécharger les données filtrées** pour télécharger les données d'audit de la plage de dates spécifiée au format CSV. Le fichier `audit.csv` est un fichier au format CSV que vous pouvez télécharger sur le système de fichiers. Vous pouvez importer ces données formatées dans un logiciel fournisseur pour mise en forme complémentaire, puis les visualiser.

Rubrique parent : Traitement des incidents

Analyse des données de journal et de trace

Vous pouvez utiliser les outils d'analyse de journal pour analyser le fonctionnement de l'environnement d'exécution et résoudre les problèmes qui y apparaissent.

Pourquoi et quand exécuter cette tâche

Vous pouvez générer des rapports à partir des fichiers journaux et de trace existants dans l'environnement. Ces rapports graphiques peuvent être utilisés pour les objectifs suivants :

- **Analyse de l'état et des performances de l'environnement d'exécution :**
  - Cohérence de l'environnement de déploiement
  - Fréquence de consignation
  - Exécution de la topologie et topologie configurée
  - Modifications non planifiées de la topologie
  - Etat de quorum
  - Etat de la réplication des partitions
  - Statistiques de mémoire, rendement, utilisation du processeur, etc.
- **Pour traiter les problèmes dans l'environnement :**
  - Vues topologiques à des points spécifiques dans le temps
  - Statistiques de mémoire, rendement, utilisation du processeur au cours des problèmes
  - Niveaux de groupe de correctifs en cours, paramètres d'optimisation
  - Etat de quorum

Présentation de l'analyse du journal
Vous pouvez utiliser l'outil **`xsLogAnalyzer`** pour vous aider traiter les problèmes dans l'environnement.

Règles de stockage des fichiers journaux

Exécution de l'analyse du journal
Vous pouvez exécuter l'outil **`xsLogAnalyzer`** sur un ensemble de fichiers journaux et de trace à partir de n'importe quel ordinateur.

Création de scanners personnalisés pour l'analyse de journal
Vous pouvez créer des scanners personnalisés pour l'analyse de journal. Après avoir configuré le scanner, les résultats sont générés dans les rapports lorsque vous exécutez l'outil **`xsLogAnalyzer`**. Le scanner personnalisé recherche les enregistrements d'événement dans les journaux en fonction des expressions régulières que vous avez définies.

Traitement des problèmes d'analyse de journal
Utilisez les informations de dépannage pour identifier et éliminer les problèmes avec l'outil **`xsLogAnalyzer`** et ses rapports générés.

Rubrique parent : Traitement des incidents

Présentation de l'analyse du journal

Vous pouvez utiliser l'outil **xsLogAnalyzer** pour vous aider traiter les problèmes dans l'environnement.

Tous les messages de reprise en ligne

Affiche le nombre total de messages de reprise en ligne sous la forme d'un graphique dans le temps. Affiche également une liste des messages de reprise en ligne, y compris les serveurs qui ont été affectés.

Tous les messages critiques eXtreme Scale

Affiche les ID de message et les explications associées et les actions utilisateur qui peuvent vous faire gagner du temps dans la recherche des messages.

Toutes les exceptions

Affiche les cinq premières exceptions, y compris les messages et leur nombre et les serveurs affectés par l'exception.

Résumé de la topologie

Affiche le diagramme de la configuration de la topologie en fonction des fichiers journaux. Vous pouvez utiliser ce récapitulatif pour comparer avec votre configuration réelle, en identifiant les erreurs de configuration.

Cohérence de topologie : tableau de comparaison ORB (Object Request Broker)

Affiche les paramètres ORB de l'environnement. Vous pouvez utiliser ce tableau pour déterminer si les paramètres de l'environnement sont cohérents.

Vue Tableau chronologique d'événements

Affiche un diagramme chronologique des différentes actions qui ont eu lieu sur la grille de données, y compris les événements de cycle de vie, les exceptions, les messages critiques et les événements de diagnostic de premier niveau (FFDC).

Rubrique parent : Analyse des données de journal et de trace

Tâches associées:
Exécution de l'analyse du journal
Création de scanners personnalisés pour l'analyse de journal
Traitement des problèmes d'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Règles de stockage des fichiers journaux

Les tableaux ci-dessous décrivent les règles de stockage des fichiers journaux sur le dispositif WebSphere DataPower XC10 Appliance.

Tableau 1. Fichier de trace xsa.app

Fichier de trace xsa.app	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	5	true
audit.log	30	2097152	5	true
*stats.log	N/A	N/A	N/A	N/A

Tableau 2. Fichier de trace xsa.admin

Fichier de trace xsa.admin	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	2	true
audit.log	30	2097152	5	true
health.log	30	2097152	2	true
*stats.log	N/A	N/A	N/A	N/A

Tableau 3. Fichier cs.trace

Fichier cs	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	2	true
audit.log	30	2097152	5	true
health.log	30	2097152	2	true
*stats.log	N/A	N/A	N/A	N/A

ts.log				

*Tableau 4. Fichier de trace xsServer**

Fichier de trace xsServer*	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	2	true
audit.log	30	2097152	5	true
health.log	30	2097152	2	true
*stats.log	N/A	104857600	10	true

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Exécution de l'analyse du journal
Création de scanners personnalisés pour l'analyse de journal
Traitement des problèmes d'analyse de journal

Exécution de l'analyse du journal

Vous pouvez exécuter l'outil **xsLogAnalyzer** sur un ensemble de fichiers journaux et de trace à partir de n'importe quel ordinateur.

Avant de commencer

- Vous devez disposer d'une installation WebSphere eXtreme Scale Client. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client.
- Collectez les fichiers journaux et de trace à partir du dispositif. Pour plus d'informations, voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance.
- Si vous voulez créer des scanners personnalisés, créez un fichier de propriétés de spécifications de scanner et un fichier de configuration avant d'exécuter l'outil. Pour plus d'informations, voir Création de scanners personnalisés pour l'analyse de journal.

Procédure

1. Exécutez l'outil **xsLogAnalyzer**.

   Le script se trouve dans les emplacements suivants :
   - Dans une installation autonome : *racine_install_wxs*/ObjectGrid/bin
   - Dans une installation intégrée à WebSphere Application Server : *racine_was*/bin

   Conseil : Si les fichiers journaux sont volumineux, utilisez les paramètres **-startTime**, **-endTime**, et **-maxRecords** lorsque vous exécutez le rapport pour limiter le nombre d'entrées de journal analysées. L'utilisation de ces paramètres lorsque vous exécutez le rapport améliore la clarté et l'exécution du rapport. Vous pouvez exécuter plusieurs rapports sur un même groupe de fichiers journaux.

   ```
   xsLogAnalyzer.sh|bat -logsRoot c:\myxslogs -outDir c:\myxslogs\out
   -startTime 11.09.27_15.10.56.089 -endTime 11.09.27_16.10.56.089 -maxRecords 100
   ```

   -logsRoot

   Spécifie le chemin absolu du répertoire des journaux à évaluer (requis).

   -outDir

   Spécifie un répertoire pour y placer la sortie du rapport. Si vous ne définissez pas une valeur, le rapport est écrit dans l'emplacement racine de l'outil **xsLogAnalyzer**.

   -startTime

   Spécifie l'heure de début de l'évaluation dans les journaux. La date est au format suivant : *année.mois.jour_heure.minute.seconde.milliseconde*

   -endTime

   Spécifie l'heure de fin de l'évaluation dans les journaux. La date est au format suivant : *année.mois.jour_heure.minute.seconde.milliseconde*

   -trace

   Spécifie une chaîne de trace, telle que `ObjectGrid*=all=enabled`.

   -maxRecords

   Spécifie le nombre maximal d'enregistrements pour générer le rapport. La valeur par défaut est `100`. Si vous définissez la valeur 50, les 50 premiers enregistrements sont générés pour la période définie.

2. Ouvrez les fichiers générés. Si vous n'avez pas défini de répertoire de sortie, les rapports sont générés dans le dossier `report_`*`date_time`*. Pour ouvrir la page principale des rapports, ouvrez le fichier `index.html`.
3. Utilisez les rapports pour analyser les données des journaux. Suivez les conseils ci-dessous pour optimiser les performances du rapport affiché :
   - Pour optimiser les performances des requêtes sur les données des journaux, utilisez des informations aussi spécifiques que possibles. Par exemple, la recherche de `server` dure plus longtemps et retourne plus de résultats que `server_host_name`.
   - Certaines vues ont un nombre limité de points de données affichés simultanément. Vous pouvez ajuster le segment de temps affiché en changeant les données en cours, telles que les heures de début et de fin, dans la vue.

Que faire ensuite

Pour plus d'informations sur le traitement de l'outil **xsLogAnalyzer** et les rapports générés, voir Traitement des problèmes d'analyse de journal.

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Création de scanners personnalisés pour l'analyse de journal
Traitement des problèmes d'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Création de scanners personnalisés pour l'analyse de journal

Vous pouvez créer des scanners personnalisés pour l'analyse de journal. Après avoir configuré le scanner, les résultats sont générés dans les rapports lorsque vous exécutez l'outil **xsLogAnalyzer**. Le scanner personnalisé recherche les enregistrements d'événement dans les journaux en fonction des expressions régulières que vous avez définies.

Procédure

1. Créez un fichier de propriétés de spécification de scanner qui définit l'expression générale à exécuter pour le scanner personnalisé.
    a. Créez et enregistrez un fichier de propriétés. Le fichier doit se trouver dans le répertoire *loganalyzer_root*/config/custom. Vous pouvez attribuer le nom de choix. Le fichier est utilisé par le nouveau scanner ; il est donc utile de nommer le scanner dans le fichier des propriétés. Par exemple, *my_new_server_scanner_spec*.properties.
    b. Incluez les propriétés suivantes dans le fichier *my_new_server_scanner_spec*.properties :

    ```
    include.regular_expression = REGULAR_EXPRESSION_TO_SCAN
    ```

    La variable *REGULAR_EXPRESSION_TO_SCAN* est une expression régulière en fonction de laquelle vous filtrez les fichiers journaux.

    Exemple : pour analyser les instances des lignes qui contiennent les chaînes "xception" et "rror", quel que soit l'ordre, affectez la valeur suivante à la propriété **include.regular_expression** :

    ```
    include.regular_expression = (xception.+rror)|(rror.+xception)
    ```

    Cette expression régulière permet d'enregistrer les événements si la chaîne "rror" se trouve avant ou après la chaîne "xception".

    Exemple :

    Pour analyser chaque ligne des journaux pour rechercher les lignes qui contiennent la chaîne "xception" ou "rror" quel que soit l'ordre, affectez la valeur suivante à la propriété **include.regular_expression** :

    ```
    include.regular_expression = (xception)|(rror)
    ```

    Cette expression régulière permet d'enregistrer les événements si la chaîne "rror" ou "xception" existe.
2. Créez un fichier de configuration que l'outil **xsLogAnalyer** utilise pour créer le scanner.
    a. Créez et enregistrez un fichier de configuration. Le fichier doit se trouver dans le répertoire *loganalyzer_root*/config/custom. Vous pouvez nommer le fichier *scanner_name*Scanner.config, où *scanner_name* est le nom unique du nouveau scanner. Par exemple, vous pouvez nommer le fichier serverScanner.config
    b. Incluez les propriétés suivantes dans le fichier *scanner_name*Scanner.config :

    ```
    scannerSpecificationFiles = LOCATION_OF_SCANNER_SPECIFICATION_FILE
    ```

    La variable *LOCATION_OF_SCANNER_SPECIFICATION_FILE* est le chemin et l'emplacement du fichier de spécification que vous avez créé au cours de l'étape précédente. Par exemple : *loganalyzer_root*/config/custom/my_new_scanner_spec.properties. Vous pouvez aussi définir plusieurs fichiers de spécification de scanner en utilisant une liste d'éléments séparés par un point-virgule :

    ```
    scannerSpecificationFiles =
    LOCATION_OF_SCANNER_SPECIFICATION_FILE1;LOCATION_OF_SCANNER_SPECIFICATION_FILE2
    ```
3. Exécutez l'outil **xsLogAnalyzer**. Pour plus d'informations, voir Exécution de l'analyse du journal.

Résultats

Après avoir exécuté l'outil **xsLogAnalyzer**, le rapport contient de nouveaux onglets pour les scanners personnalisés que vous avez configurés. Chaque onglet contient les vues suivantes :

Graphiques

Graphique qui illustre les événements enregistrés. Les événements sont affichés dans leurs ordre de découverte.

Tableaux

Représentation tabulaire des événements enregistrés.

Etats récapitulatifs

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Exécution de l'analyse du journal
Traitement des problèmes d'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Traitement des problèmes d'analyse de journal

Utilisez les informations de dépannage pour identifier et éliminer les problèmes avec l'outil **xsLogAnalyzer** et ses rapports générés.

Procédure

- **Problème :** manque de mémoire lors de l'utilisation de l'outil **xsLogAnalyzer** pour générer des rapports. Exemple d'erreur possible : `java.lang.OutOfMemoryError: GC overhead limit exceeded`.

  Solution : l'outil **xsLogAnalyzer** s'exécute dans une machine JVM (Java virtual machine). Vous pouvez configurer la machine JVM pour augmenter la taille de segment avant d'exécuter l'outil **xsLogAnalyzer** en définissant certains paramètres lorsque vous exécutez l'outil. L'augmentation de la taille du segment permet de stocker plus d'enregistrements dans la mémoire JVM. Commencez avec 2 048 M en supposant que le système d'exploitation dispose d'une mémoire principale suffisante. Dans la même instance de ligne de commande dans laquelle vous voulez exécuter l'outil **xsLogAnalyzer**, définissez la taille de segment de mémoire JVM maximale :

  ```
  java -Xmx<HEAP_SIZE>m
  ```

  La valeur *HEAP_SIZE* peut être un entier et représente le nombre de mégaoctets alloués au segment de mémoire JVM.
  Par exemple, vous pouvez exécuter `java -Xmx2048m`. Si vous continuez de recevoir des messages indiquant un manque de mémoire ou que vous ne disposez pas des ressources pour allouer 2 048 Mo ou plus, limitez le nombre d'événements stockés dans le segment de mémoire. Vous pouvez limiter le nombre d'événements dans le segment de mémoire en envoyant le paramètre **-maxRecords** dans la commande **xsLogAnalyzer**.

- **Problème :** lorsque vous ouvrez un rapport généré à partir de l'outil **xsLogAnalyzer**, le navigateur se bloque et ne charge pas la page.

  Cause : les fichiers HTML générés sont trop volumineux et le navigateur ne peut pas les charger. Ces fichiers sont volumineux, car la portée des fichiers journaux que vous analysez est trop grande.

  Solution : utilisez les paramètres **-startTime**, **-endTime**, et **-maxRecords** lorsque vous exécutez l'outil **xsLogAnalyzer** pour limiter le nombre d'entrées de journal analysées. L'utilisation de ces paramètres lorsque vous exécutez le rapport améliore la clarté et l'exécution du rapport. Vous pouvez exécuter plusieurs rapports sur un même groupe de fichiers journaux.

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Exécution de l'analyse du journal
Création de scanners personnalisés pour l'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Surveillance de la température du matériel

Des capteurs mesurent en permanence la température de divers composants internes du dispositif. Ces températures peuvent être surveillées facilement à partir d'un seul panneau à l'aide de l'interface utilisateur.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Le système IBM® WebSphere DataPower XC10 Appliance surveille la température interne de divers composants pour s'assurer que le dispositif fonctionne à une température raisonnable.

Procédure

1. Accédez à l'option **Dispositif** > **Identification et résolution des incidents**.
2. Développez l'entrée **Températures du matériel**.
3. Examinez les températures de votre matériel

   Le relevé le plus récent de chaque capteur est affiché avec trois chiffres significatifs.
   - Si la température du composant correspond à une température de fonctionnement normale, une icône verte () est affichée.
   - Si la température du composant dépasse le niveau de sécurité, une icône d'avertissement jaune () est affichée.

Résultats

A l'issue de ces étapes, vous aurez examiné les relevés de températures disponibles pour vous assurer que votre dispositif fonctionne à une température raisonnable.

Que faire ensuite

Continuez à résoudre l'incident auquel vous êtes confronté.

Rubrique parent : Traitement des incidents

Surveillance du statut des interfaces Ethernet à l'aide d'une connexion série

Si votre système IBM® WebSphere DataPower XC10 Appliance rencontre des incidents au niveau des réseaux, vous pouvez afficher des informations sur l'activité et le statut des interfaces Ethernet. Cette rubrique décrit comment afficher ces informations à l'aide d'une connexion série établie avec le dispositif. Notez que vous pouvez aussi afficher ces informations au moyen de l'interface utilisateur.

Avant de commencer

Vous devez avoir un accès physique au dispositif et vous connecter avec le compte xcadmin.

Procédure

1. Etablissez une connexion série avec le dispositif en tant qu'utilisateur `xcadmin`.

   La connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif. Si vous utilisez un PC comme console série, vous devez utiliser un programme de communication série basé sur PC Windows ou Linux. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion avec le dispositif.

   Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de pouvoir continuer.

   Les paramètres de la console série sont *9600.8.n.1*.
2. Affichez le statut de l'interface Ethernet. Utilisez la commande suivante pour afficher le statut de l'interface Ethernet :

   ```
   netif status <interface>
   ```

   Où <*interface*> correspond au nom de l'interface Ethernet à interroger. Vous trouverez ci-après un exemple de sortie :

   ```
   Console> netif status mgt0
   mgt0    generic MTU:1500 flags:UP BROADCAST RUNNING MULTICAST
           inet addr:127.0.0.1 mask:255.255.255.128
           inet6 addr: 2001:DB8:0:0:0:0:0:0/32 mask: ffff:ffff:ffff:ffff::
           ethernet MAC: 00:1a:64:88:a0:6c autoneg:on duplex:Full port:TP
               speed:100Mbps
           statistics collisions:0 multicast:13 rx_bytes:101069093
               rx_compressed:0 rx_crc_errors:0 rx_dropped:0 rx_errors:0
               rx_fifo_errors:0 rx_frame_errors:0 rx_length_errors:0
               rx_missed_errors:0 rx_over_errors:0 rx_packets:1515625
               tx_aborted_errors:0 tx_bytes:10045272 tx_carrier_errors:0
               tx_compressed:0 tx_dropped:0 tx_errors:0 tx_fifo_errors:0
               tx_heartbeat_errors:0 tx_packets:20104 tx_window_errors:0
   ```

Résultats

A l'issue des ces étapes, vous avez consulté le détail du statut des interfaces Ethernet de votre dispositif.

Rubrique parent : Traitement des incidents

Vérification des connexions sortantes du dispositif

A l'aide de la fonction Connexion sortante, vous pouvez vérifier si une adresse réseau est accessible à partir du dispositif.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

La fonction de connexion sortante peut aider à isoler un problème en vue de le résoudre en confirmant qu'une adresse réseau cible est accessible à partir du dispositif. Cet outil n'est pas forcément pertinent au débogage de chaque incident mais constitue une méthode pratique pour s'assurer qu'une adresse réseau cible est disponible et que la connectivité avec cette adresse n'est pas entravée par un pare-feu ou par un problème réseau.

Procédure

1. Accédez à l'option **Dispositif** > **Identification et résolution des incidents**.
2. Développez l'entrée **Connexions sortantes**.
3. Entrez une adresse IP ou un nom d'hôte complet. L'adresse réseau saisie dans cette zone est utilisée en tant qu'adresse cible lorsque la commande ping est émise.
4. Cliquez sur **Commande Ping** pour lancer cette commande de sondage sur l'adresse réseau indiquée.
    - Si la tentative de connexion à l'adresse réseau saisie aboutit, l'icône suivante est affichée : .
    - Si la tentative de connexion à l'adresse réseau saisie n'a pas abouti, l'icône suivante est affichée : .

Résultats

A l'issue de ces étapes, vous aurez déterminé si une adresse cible est actuellement accessible sur le réseau à partir du dispositif.

Que faire ensuite

Continuez à résoudre l'incident auquel vous êtes confronté.

Rubrique parent : Traitement des incidents

Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif

Vous pouvez exécuter des commandes pour redémarrer le matériel du dispositif, rétablir les paramètres usine du dispositif ou arrêter le dispositif.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'interface de ligne de commande pour réinitialiser les données de configuration du dispositif, redémarrer le dispositif ou l'arrêter.

Procédure

1. Etablissez une connexion avec le dispositif en utilisant l'utilisateur `xcadmin`. Vous pouvez utiliser une connexion série ou vous connecter via l'interface de ligne de commande à l'aide d'un shell sécurisé (SSH).
    - **Connexion série :** La connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif. Si vous utilisez un PC comme console série, vous devez utiliser un programme de communication série basé sur PC pour Windows ou Linux. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion avec le dispositif.

      Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de continuer.
    - **Connexion éloignée :** Pour vous connecter au dispositif à l'aide de SSH, spécifiez l'URL de votre dispositif à votre client SSH.
2. Une fois la connexion avec le dispositif établie, vous pouvez exécuter des commandes.

   2.5+ Pour obtenir des informations de référence concernant les commandes que vous pouvez exécuter, voir Références des commandes de l'interface CLI.

Que faire ensuite

Si vous reconfigurez le dispositif, vous pouvez le réinitialiser. Pour plus d'informations sur la configuration du dispositif, reportez-vous à la rubrique Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Traitement des incidents

Tâches associées:
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Contrôle des activités à l'aide des tâches

Traitement des problèmes d'interblocage

Cette rubrique décrit des scénarios d'interblocage courants et des solutions pour les éviter.

Avant de commencer

Mettez en oeuvre la gestion des exceptions dans votre application. Pour plus d'informations, voir Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java et Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET.

L'exception suivante s'affiche :

```
com.ibm.websphere.objectgrid.plugins.LockDeadlockException: Message
```

Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

Procédure

- **Problème :** Une exception LockTimeoutException se produit.

  Description : Lorsqu'une transaction ou un client demande qu'un verrou soit octroyé pour une entrée de mappe spécifique, le système attend généralement que le client en cours libère le verrou avant d'envoyer la demande. Si la demande de verrou reste inactive pendant une longue période et qu'aucun verrou n'est jamais accordé, l'exception LockTimeoutException est créée pour empêcher un interblocage, ce qui est décrit plus en détail dans la section suivante. Il est plus probable que vous receviez cette exception lorsque vous configurez une stratégie de verrouillage pessimiste, car le verrou n'est jamais libéré avant la validation de la transaction.

  Extraire plus de détails :

  - Java L'exception LockTimeoutException contient la méthode getLockRequestQueueDetails, qui renvoie une chaîne. Vous pouvez utiliser cette méthode pour visualiser la description détaillée de la situation qui déclenche l'exception. Voici un exemple de code qui intercepte l'exception et affiche un message d'erreur :

    ```
    try {
        ...
    }
    catch (LockTimeoutException lte) {
        System.out.println(lte.getLockRequestQueueDetails());
    }
    ```

    Si vous recevez l'exception dans un bloc catch d'exception ObjectGridException, le code ci-dessous détermine l'exception et affiche les détails de la file d'attente. Il utilise également la méthode de l'utilitaire findRootCause.

    ```
    try {
    ...
    }
    catch (ObjectGridException oe) {
        Throwable Root = findRootCause( oe );
        if (Root instanceof LockTimeoutException) {
            LockTimeoutException lte = (LockTimeoutException)Root;
            System.out.println(lte.getLockRequestQueueDetails());
        }
    }
    ```

  - .NET L'exception LockTimeoutException contient la méthode getMessage, qui renvoie une chaîne. Vous pouvez utiliser cette méthode pour visualiser la description détaillée de la situation qui déclenche l'exception.

  Solution : une exception LockTimeoutException empêche les blocages possibles dans votre application. Une exception de ce type se produit lorsque l'application attend pendant un laps de temps défini. Vous pouvez définir la durée maximum pendant laquelle l'application attend en définissant un délai de verrouillage. Si aucun interblocage réel n'existe dans l'application, ajustez le délai d'attente de verrouillage pour éviter l'exception LockTimeoutException.

  Configurer le délai de verrouillage à l'aide d'un programme.

  - Java Configuration et mise en oeuvre du verrouillage dans les applications Java

    - .NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

.

- **Problème :** Un interblocage se produit sur une clé unique.

  Description : Les scénarios suivants décrivent comment des interblocages peuvent se produire lors de l'accès à une clé unique avec un verrou S et que cette clé est mise à jour ultérieurement. Lorsque deux transactions effectuent simultanément cette action, un interblocage se produit.

  Tableau 1. Scénario d'interblocage sur clé unique

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1	get clé1	Verrou S octroyé aux deux transactions pour clé1
3	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1		Aucun verrou U. Mise à jour effectuée dans le cache transactionnel.
4		L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	Aucun verrou U. Mise à jour effectuée dans le cache transactionnel
5	Validation de la transaction		Bloquée : le verrou S pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.
6		Validation de la transaction	Interblocage : le verrou S pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 1 détient un verrou S.

  Tableau 2. Interblocages sur clé unique (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1		Verrou S octroyé pour clé1
3	L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	get clé1	Verrou S mis à niveau vers un verrou U pour clé1.
4		get clé1	Verrou S octroyé pour clé1
5		L'une des deux opérations suivantes : ◦ Java getForUpdate	Bloquée : T1 détient déjà un verrou U.

		• clé1 • .NET GetAndLock clé1	
6	Validation de la transaction		Interblocage : le verrou U pour clé1 ne peut pas être mis à niveau.
7		Validation de la transaction	Interblocage : le verrou S pour clé1 ne peut pas être mis à niveau.

Tableau 3. Interblocages sur clé unique (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1		Verrou S octroyé pour clé1
3	L'une des deux opérations suivantes : • Java getForUpdate clé1 • .NET GetAndLock clé1		Verrou S mis à niveau vers un verrou U pour clé1.
4		get clé1	Verrou S octroyé pour clé1
5		L'une des deux opérations suivantes : • Java getForUpdate clé1 • .NET GetAndLock clé1	Bloquée : unité d'exécution 1 détient déjà un verrou U.
6	Validation de la transaction		Interblocage : le verrou U pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.

Si ObjectMap.getForUpdate est utilisée pour éviter le verrou S, l'interblocage est évité :

Tableau 4. Interblocages sur clé unique (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	L'une des deux opérations suivantes : • Java getForUpdate clé1 • .NET GetAndLock clé1		Verrou U octroyé à l'unité d'exécution 1 pour clé1.
3		L'une des deux opérations suivantes :	Demande de verrou U bloquée.

		◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	
4	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	<bloquée>	
5	Validation de la transaction	<bloquée>	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.
6		<libérée>	Le verrou U est finalement octroyé à l'unité d'exécution 2 pour clé1.
7		L'une des deux opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	Verrou U octroyé à l'unité d'exécution 2 pour clé2.
8		Validation de la transaction	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.

Solutions :

- Utilisez la méthode getForUpdate ou GetAndLock au lieu de get pour obtenir un verrou U au lieu d'un verrou S.
- Utilisez le niveau d'isolement de transaction lecture validée pour éviter de maintenir des verrous S. La réduction du niveau d'isolement de transaction augmente le risque de lectures non reproductibles. Toutefois, les lectures non reproductibles à partir d'un client ne sont possibles que si le cache transactionnel est explicitement invalidé par le même client.
- Java Utilisez la stratégie de verrouillage optimiste. L'utilisation de cette stratégie requiert le traitement des exceptions de conflits optimistes. (Applications Java uniquement)

- **Problème :** Un interblocage se produit sur plusieurs clés ordonnées.

  Description : Ce scénario décrit ce qui se produit si deux transactions tentent de mettre à jour directement la même entrée et maintiennent des verrous S sur d'autres entrées.

  Tableau 5. Scénario d'interblocage sur plusieurs clés

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1	get clé1	Verrou S octroyé aux deux transactions pour clé1
3	get clé2	get clé2	Verrou S octroyé aux deux transactions pour clé2
4	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1		Aucun verrou U. Mise à jour effectuée dans le cache transactionnel.
5		L'une des deux	Aucun verrou U. Mise à jour effectuée dans le cache transactionnel.

		opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	
6.	Validation de la transaction		Bloquée : le verrou S pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.
7		Validation de la transaction	Bloquée : le verrou S pour clé2 ne peut pas être mis à niveau car l'unité d'exécution 1 détient un verrou S.

La méthode ObjectMap.getForUpdate permet d'éviter le verrou S, et donc l'interblocage :

Tableau 6. Scénario d'interblocage sur clés multiples (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1		Verrou U octroyé à la transaction T1 pour clé1.
3		L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	Demande de verrou U bloquée.
4	get clé2	<bloquée>	Verrou S octroyé à la transaction T1 pour clé2.
5	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	<bloquée>	
6	Validation de la transaction	<bloquée>	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.
7		<libérée>	Le verrou U est finalement octroyé à la transaction T2 pour clé1.
8		get clé2	Verrou S octroyé à la transaction T2 pour clé2
9		L'une des deux opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	Verrou U octroyé à la transaction T2 pour clé2
10		Validation de la transaction	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.

Solutions :

- Utilisez la méthode getForUpdate ou GetAndLock au lieu de la méthode get pour acquérir directement un verrou U pour la première clé. Cette stratégie ne fonctionne que si l'ordre des méthodes est déterministe.
- Utilisez le niveau d'isolement de transaction lecture validée pour éviter de maintenir des verrous S. Il s'agit de la solution la plus simple à mettre en oeuvre si l'ordre des méthodes n'est pas déterministe. La réduction du niveau d'isolement de transaction augmente le risque de lectures non reproductibles. Toutefois, les lectures non reproductibles ne sont possibles que si le cache transactionnel est explicitement invalidé.
- Java Utilisez la stratégie de verrouillage optimiste. L'utilisation de cette stratégie requiert le traitement des exceptions de conflits optimistes. (Applications Java uniquement.)

- **Problème :** Un interblocage se produit à cause d'un verrou U mal ordonné.

  Description : Si l'ordre dans lequel les clés sont demandées ne peut pas être garanti, un interblocage peut toujours se produire.

Tableau 7. Scénario d'erreur d'ordre pour le verrou U

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	L'une des deux opérations suivantes : ◦ Java getForUpdate clé2 ◦ .NET GetAndLock clé2	Verrou U octroyé pour clé1 et clé2
3	get clé2	get clé1	Verrou S octroyé pour clé1 et clé2
4	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	L'une des deux opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	
5	Validation de la transaction		Le verrou U ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.
6		Validation de la transaction	Le verrou U ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 1 détient un verrou S.

Solutions :

- Effectuez tout le travail avec un verrou U simple global (mutex). Cette méthode réduit les possibilités d'accès simultané mais gère tous les scénarios lorsque l'accès et l'ordre ne sont pas déterministes.
- Utilisez le niveau d'isolement de transaction lecture validée pour éviter de maintenir des verrous S. Il s'agit de la solution la plus simple à mettre en oeuvre et qui offre le plus de possibilités d'accès simultané lorsque l'ordre des méthodes n'est pas déterministe. La réduction du niveau d'isolement de transaction augmente le risque de lectures non reproductibles. Toutefois, les lectures non reproductibles ne sont possibles que si le cache transactionnel est explicitement invalidé.
- Java Utilisez la stratégie de verrouillage optimiste. L'utilisation de cette stratégie requiert le traitement des exceptions de conflits optimistes. (Applications Java uniquement.)

Rubrique parent : Traitement des incidents

Java

Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition

2.5+ Le scénario décrit est un exemple de transaction multipartition qui est à l'origine d'une exception de dépassement de délai de verrouillage. Selon l'état de la transaction, les solutions montrent la façon dont vous pouvez manuellement résoudre ce problème.

Avant de commencer

Implémentez la gestion des exceptions dans l'application. Pour plus d'informations, voir Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java.

L'exception suivante s'affiche :

```
Caused by: com.ibm.websphere.objectgrid.LockTimeoutException: Local-40000139-DEF8-05EA-
E000-64A856931719 timed out waiting for lock mode S to be granted for map name: TS2_MapP,
key: key12
granted = X
lock request queue
->[WXS-40000139-DEF6-FA84-E000-1CB456931719, state = Granted, requested 73423 milli-
seconds ago, marked to keep current mode false, snapshot mode 0, mode = X, thread name =
xIOReplicationWorkerThreadPool : 29]
->[Local-40000139-DEF8-05EA-E000-64A856931719, state = Waiting for 5000 milli-seconds,
marked to keep current mode false, snapshot mode 0, mode = S, thread name =
xIOWorkerThreadPool : 28]
dump of all locks for WXS-40000139-DEF6-FA84-E000-1CB456931719
Key: key12, map: TS2_MapP
strongest currently granted mode for key is X
->[WXS-40000139-DEF6-FA84-E000-1CB456931719, state = Granted, requested 73423 milli-
seconds ago, marked to keep current mode false, snapshot mode 0, mode = X, thread name =
xIOReplicationWorkerThreadPool : 29]
dump of all locks for Local-40000139-DEF8-05EA-E000-64A856931719
```

Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

Procédure

Problème : vous obtenez une exception de dépassement de délai de verrouillage et le détenteur du verrou est une transaction multipartition, ou le dossier du journal augmente avec les messages de journal.

Diagnostic :

un message apparaît de manière répétée jusqu'à remplir le dossier des journaux, par exemple :

```
00000099 TransactionLog I  CWOBJ8705I: Automatic resolution of transaction WXS-40000139-
DF01-216D-E002-1CB456931719 at RM:TestGrid:TestSet2:20 is still waiting for a decision.
Another attempt to resolve the transaction will occur in 30 seconds.
```

Identifiez le type de transaction à l'origine du verrou. Si le préfixe dans l'identificateur de transaction est WXS-, cela indique qu'il s'agit d'une transaction multipartition. Si le préfixe dans l'identificateur de transaction est Local-, cela indique qu'il s'agit d'une transaction à une partition.

Cause : il est fort probable que l'application détienne le verrou, car aucune validation ou annulation n'a eu lieu.

Solution : déterminez l'état de la transaction et la durée de l'état. Utilisez l'utilitaire de commande **`xscmd -c listindoubts`** avec l'option `-d` (pour une sortie détaillée) ou le bean géré de la transaction.

> 2.5+ **Résolution des exceptions de délai d'attente de verrouillage**
> En utilisant la commande **`xscmd -c listindoubt`**, vous pouvez afficher l'état d'une transaction et déterminer l'action à exécuter.

Rubrique parent : Traitement des incidents

Concepts associés:
Java Stratégies de verrouillage
Java Validation en deux phases et reprise sur incident

Tâches associées:

Java Résolution des exceptions de délai d'attente de verrouillage

Java

Résolution des exceptions de délai d'attente de verrouillage

2.5+ En utilisant la commande **`xscmd -c listindoubt`**, vous pouvez afficher l'état d'une transaction et déterminer l'action à exécuter.

Rubrique parent : 2.5+ Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition

Concepts associés:
Java Stratégies de verrouillage
Java Validation en deux phases et reprise sur incident

Tâches associées:
Java Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition

Résolution des exceptions de délai d'attente de verrouillage à l'aide de la commande `xscmd -c listindoubts`

Procédure

- Affichez la liste détaillée des transactions dans votre environnement : `xscmd -c listindoubt -d`
- Exécutez les actions appropriées pour résoudre la transaction. **Problème :** La transaction est marquée comme validée à TM mais les RM sont en attente de validation.

```
[1] WXS-40000139-DEF8-EF60-E002-1CB456931719
Timestamp                 Partition    Role  State     Container   Resync  Attempts
-------------------------------------------------------------------------------
--
2012-09-19 10:40:19.824  TestSet1:11  TM   COMMIT    MPTBasic2_C-0   Primary  0
2012-09-19 10:40:19.824         TestSet1:7      RM      PREPARED        MPTBasic0_C-
1       Primary         0
2012-09-19 10:40:19.839         TestSet2:20     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:40:19.824  TestSet2:6   RM   PREPARED  MPTBasic0_C-1   Primary  0
```

Solution : valider les partitions de gestionnaire de ressources (RM) et ignorez la transaction.

1. Exécutez la commande suivante pour valider la partition RM dans la transaction `WXS-40000139-DEF8-EF60-E002-1CB456931719`: `xscmd -c listIndoubts -xid WXS-40000139-DEF8-EF60-E002-1CB456931719 -cm -rm`
2. Exécutez la commande suivante pour ignorer cette transaction : `xscmd -c listIndoubts -xid WXS-40000139-DEF8-EF60-E002-1CB456931719 -f`

Problème : La transaction est en attente de validation sur toutes les partitions.

```
[1] WXS-40000139-DEF6-FA84-E000-1CB456931719
Timestamp                 Partition    Role  State     Container   Resync  Attempts
-------------------------------------------------------------------------------
--
2012-09-19 10:38:11.603         TestSet1:10     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:38:11.588         TestSet1:5      TM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:38:11.603         TestSet2:11     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:38:11.619  TestSet2:13  RM   PREPARED  MPTBasic2_C-0   Primary  0
```

Solution : annulez la partition TM et les partitions RM suivantes. Ensuite ignorez la transaction.

1. Exécutez la commande suivante pour annuler la partition TM dans la transaction `WXS-40000139-DEF6-FA84-E000-1CB456931719`: `xscmd -c listIndoubts -xid WXS-40000139-DEF6-FA84-E000-1CB456931719 -r -tm`
2. Exécutez la commande suivante pour annuler les partitions RM dans cette transaction : `xscmd -c listIndoubts -xid WXS-40000139-DEF6-FA84-E000-1CB456931719 -r -rm`
3. Exécutez la commande suivante pour ignorer cette transaction : `xscmd -c listIndoubts -xid WXS-40000139-DEF6-FA84-E000-1CB456931719 -f`

Problème : La transaction est en attente de validation sur toutes les partitions RM, mais la décision

de transaction est inconnue au niveau de TM.

```
[1] WXS-40000139-DEF8-EF31-E000-1CB456931719
Timestamp                 Partition    Role  State     Container   Resync  Attempts
-------------------------------------------------------------------------------
2012-09-19 10:40:19.777         TestSet1:11     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:40:19.792         TestSet2:5      RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:40:19.777 TestSet2:6   RM    PREPARED MPTBasic2_C-1 Primary  0
```

Solution : annulez les partitions RM.

- Exécutez la commande suivante pour annuler les partitions RM dans la transaction `WXS-40000139-DEF8-EF31-E000-1CB456931719` : `xscmd -c listIndoubts -xid WXS-40000139-DEF8-EF31-E000-1CB456931719 -r`

Traitement des problèmes d'intégration du cache

Utilisez ces informations pour traiter les problèmes de la configuration de l'intégration du cache, y compris ceux associés aux configurations de session HTTP et de cache dynamique.

Procédure

- **Problème :** les ID de session HTTP ne sont pas réutilisés.

  Cause : vous pouvez réutiliser les ID de session. Si vous créez une grille de données pour la persistance des sessions dans la version 7.1.1 ou une version ultérieure, la réutilisation des ID de session est automatiquement activée. Toutefois, si vous avez créé des configurations dans des versions antérieures, ce paramètre est peut être déjà défini avec une valeur incorrecte.

  Solution : vérifiez les paramètres suivants pour déterminer si vous avez activé la réutilisation des ID de session HTTP :
  - La propriété `reuseSessionId` dans le fichier `splicer.properties` doit avoir la valeur `true`.
  - La propriété personnalisée `HttpSessionIdReuse` doit avoir la valeur `true`. Cette propriété personnalisée peut être définie dans l'un des chemins suivants dans la console d'administration WebSphere Application Server :
    - **Serveurs** > ***server_name*** > **Getion de session** > **Propriétés personnalisées**
    - **Clusters dynamiques** > ***dynamic_cluster_name*** > **Modèle de serveur** > **Gestin de session** > **Propriétés personnalisés**
    - **Serveurs** > **Types de serveur** > **Serveurs d'applications WebSphere** > ***server_name***, puis sous Infrastructure du serveur, cliquez sur **Java et gestion des processus** > **Définition de processus** > **Java virtual machine** > **Propriétés personnalisées**
    - **Serveurs** > **Types de serveur** > **Serveurs d'applications WebSphere** > ***server_name*** > **Paramètres de conteneur Web** > **Conteneur Web**

  Si vous mettez à jour les valeurs des propriétés personnalisées, reconfigurez la gestion des sessions eXtreme Scale afin que le fichier `splicer.properties` détecte la modification

- **Problème :** lorsque vous utilisez un grille de données pour stocker les sessions HTTP et que la charge des transactions est élevée, le message CWOBJ0006W figure dans le fichier `SystemOut.log`.

  ```
  CWOBJ0006W: An exception occurred:
  com.ibm.websphere.objectgrid.ObjectGridRuntimeException:
  java.util.ConcurrentModificationException
  ```

  Ce message apparaît lorsque le paramètre l'application Web modifie l'objet List défini comme attribut dans la session HTTPSession.

  Solution : clonez l'attribut qui contient l'objet List modifié et placez l'attribut cloné dans l'objet session.

- **Problème :** lors de l'exécution des applications Web avec la spécification Servlet 3,0, les filtres d'application Web et les programmes d'écoute ne sont pas appelés par la gestion de session WebSphere eXtreme Scale. Par exemple, les programmes d'écoute ne sont pas rappelés lorsque les sessions sont invalidées à l'aide d'explusion conteneur distant avec WebSphere eXtreme Scale.

  Cause : WebSphere eXtreme Scale n'identifie pas les filtres ni les programmes d'écoute définis en utilisant des annotations ou un programme.

  Solution : les filtres et programmes d'écoute doivent être explicitement déclarés dans le fichier web.xml de l'application Web.

Rubrique parent : Traitement des incidents

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Traitement des problèmes d'installation

Utilisez ces informations pour traiter les problèmes liés à l'installation et aux mises à jour.

Procédure

- **Problème :** lorsque vous exécutez la commande d'installation à partir d'un ordinateur distant, tel que `\\mymachine\downloads\`, le message suivant s'affiche : `CMD.EXE was started with the above path as the current directory. UNC paths are not supported. Defaulting to Windows directory.` En conséquence, l'installation ne peut pas se terminer correctement.

  Solution : mappez l'ordinateur distant à une unité réseau. Par exemple, dans Windows, vous pouvez cliquer avec le bouton droit de la souris sur **Ordinateur**, choisir **Connecter un lecteur réseau** et inclure le chemin UNC (uniform naming conventions) vers l'ordinateur distant. Ensuite, vous pouvez exécuter le script d'installation à partir du lecteur réseau avec succès. Par exemple, `y:\mymachine\downloads\WXS\install.bat`.

- **Problème :** l'installation n'aboutit pas.

  Solution : vérifiez les fichiers journaux pour savoir où l'installation a échoué. Lorsque l'installation n'aboutit pas, les fichiers journaux se trouvent dans le répertoire *racine_install_wxs*`/logs/wxs`.

- **Problème :** échec catastrophique lors de l'installation.

  Solution : vérifiez les fichiers journaux pour savoir où l'installation a échoué. Lorsque l'installation a été partiellement exécutée, le journaux se trouvent généralement dans le répertoire *user_root*`/wxs_install_logs/`.

- Windows **Problème :** si vous installez WebSphere eXtreme Scale Client sur Windows, le texte suivant peut s'afficher dans les résultats de l'installation :

  ```
  Success: The installation of the following product was successful:
  WebSphere eXtreme Scale Client.  Some configuration steps have errors.
  For more information, refer to the following log file:
  <WebSphere Application Server install root>\logs\wxs_client\install\log.txt"
  Review the installation log (log.txt) and review the deployment manager
  augmentation log.
  ```

  Solution : si vous identifiez une erreur avec le fichier `iscdeploy.sh`, vous pouvez l'ignorer. Cette erreur ne pose pas de problème.

- Linux **Problème :**

  Si vous disposez d'une installation complète et que vous tentez d'appliquer uniquement la maintenance WebSphere eXtreme Scale Client à l'aide du programme d'installation de mises à jour, le message suivant apparaît :

  ```
  Prerequisite checking has failed. Click Back to select a different package, or click
  Cancel to exit.

  Failure messages are:

  Required feature wxs.client.primary is not found.
  ```

  Si WebSphere eXtreme Scale Client est installé et que vous tentez d'appliquer un package de maintenance complet à l'aide du programme d'installation de mises à jour, le message suivant apparaît :

  ```
  Prerequisite checking has failed. Click Back to select a different package, or click
  Cancel to exit.

  Failure messages are:

  Required feature wxs.primary is not found.
  ```

  Solution : Le package de maintenance que vous installez doit correspondre au type d'installation. Téléchargez et appliquez le package de maintenance qui s'applique à votre type d'installation.

- Linux **Problème :** l'installation se bloque.

  Solution : Parfois, lors de l'installation de WebSphere eXtreme Scale sous Linux en tant qu'utilisateur

non superutilisateur, le programme d'installation peut se bloquer. Cela est probablement du au fait que le nombre maximal de fichiers ouverts est défini sur une valeur trop basse sur votre système d'exploitation Linux. Vous devez augmenter la limite autorisée dans le fichier `/etc/limits.conf` or `/etc/security/limits.conf` (l'emplacement du fichier dépend de votre distribution Linux spécifique) sur la valeur minimum 8192.

Rubrique parent : Traitement des incidents

Tâches associées:
Utilisation du programme d'installation de mises à jour pour installer des packages de maintenance

Traitement des problèmes d'administration

Utilisez les informations suivantes pour traiter les problèmes d'administration, notamment le démarrage et l'arrêt des serveurs, en utilisant l'utilitaire **xscmd**, etc.

Procédure

- **Problème :** lorsque vous exécutez une commande **xscmd**, le message suivant s'affiche :

  ```
  java.lang.IllegalStateException: Placement service MBean not available.
  []
          at
  com.ibm.websphere.samples.objectgrid.admin.OGAdmin.main(OGAdmin.java:1449)
          at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
          at
  sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
          at
  sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37
  )
          at java.lang.reflect.Method.invoke(Method.java:611)
          at com.ibm.ws.bootstrap.WSLauncher.main(WSLauncher.java:267)
  Ending at: 2011-11-10 18:13:00.000000484
  ```

  Cause :problème de connexion avec le serveur de catalogue.

  Solution : vérifiez que les serveurs de catalogue sont actifs et disponibles via le réseau. Ce message peut aussi être généré lorsque vous disposez d'un domaine de service de catalogue défini et que moins de deux serveurs de catalogue sont actifs. L'environnement n'est pas disponible tant que deux serveurs de catalogue ne sont pas démarrés.

- **Problème :** lorsque vous exécutez une commande **xscmd**, le message suivant s'affiche :

  ```
  CWXSI0066E: Unmatched argument argument_name was detected.
  ```

  Cause : Vous avez entré un format de commande non reconnu par l'utilitaire **xscmd**.

  Solution : Vérifiez le format de la commande. Ce produit peut survenir lors de l'exécution d'expressions régulières avec la commande **-c findbyKey**. Pour plus d'informations, voir Demande, affichage et invalidation des données.

Rubrique parent : Traitement des incidents

Référence

Les informations de référence sont organisées pour vous aider à rechercher rapidement des faits.

Référence à l'utilitaire xscmd
Vous pouvez utiliser la liste de commandes suivante comme référence lorsque vous utilisez l'utilitaire **xscmd**.

2.5+ **Guide de référence de l'interface de commande HTTP**

Fichier de propriétés du client
Création d'un fichier de propriétés en fonction de vos exigences pour les processus client WebSphere eXtreme Scale.

Syntaxe d'expression régulière
Une expression régulière est une chaîne structurée utilisée pour la mise en correspondances d'autres chaînes. Vous pouvez utiliser des expressions régulières pour les données d'invalidation et pour le filtrage des messages qui s'affichent dans Message Center.

Options de configuration de mappe dynamique
Vous pouvez créer des mappes supplémentaires dans une grille de données en demandant à votre application client de se connecter à la mappe spécifiquement nommée. Une fois cette connexion établie, la mappe est automatiquement créée.

Paramètres de l'interface utilisateur
Ces informations de référence décrivent les paramètres que vous pouvez afficher et configurer dans les pages de la console d'administration WebSphere Application Server et ailleurs.

Référence à l'utilitaire `xscmd`

Vous pouvez utiliser la liste de commandes suivante comme référence lorsque vous utilisez l'utilitaire **`xscmd`**.

Paramètres de commande généraux

La syntaxe générale des commandes de l'utilitaire **`xscmd`** est la suivante. Les paramètres entre crochets sont facultatifs [ ].

ATTENTION :

N'utilisez pas les commandes suivantes dans un environnement WebSphere DataPower XC10 Appliance :

- **-c releaseShard**
- **-c reserveShard**
- **-c swapShardWithPrimary**
- **-c suspendBalancing**
- **-c resumeBalancing**
- **-c teardown**
- **-c triggerPlacement**
- **-c enableForPlacement**

```
Syntaxe : xscmd -c <cmdName> | -h <cmdName> | -lc
[<cmdGroupName>] | -lcg
             [-tt <type>] [-prot <protocol>] [-trf <filePath>] [-ks
             <filePath>] [-ksp <password>] [-user <username>] [-al <alias>]
             [-cgp <property>] [-kst <type>] [-cep <endpoints>] [-ssp
             <profileName>] [-tsp <password>] [-arc <integer>] [-trs
             <traceSpec>] [-tst <type>] [-to <serverTimeout>] [-cxpv
             <provider>] [-sp <profileName>] [-pwd <password>] [-ca <support>]
             [-cgc <className>] [-ts <filePath>]

Options:
  -al,--alias <alias>                  Nom d'alias dans le fichier de clés.

  -arc,--authRetryCount <integer>        Nombre de tentatives d'authentification si
                                       les données d'identification ont expiré. Si la
                                       valeur est 0, aucune autre tentative
d'authentification
                                       n'est effectuée.

  -c,--command <cmdName>                 Définit le nom de la commande à
                                       exécuter

  -ca,--credAuth <support>             Définit le support d'authentification des
                                       données d'identification du client [Never,
                                       Supported, Required].

  -cep,--catalogEndPoints <endpoints>    Définit un ou plusieurs noeuds finaux de service
de catalogue
                                       dans le format
                                       <host>[:<listenerPort>][,<host>[:
                                       <listenerPort>]]. Noeud final par défaut :
                                       localhost:2809

  -cgc,--credGenClass <className>        Spécifie le nom de la classe qui
                                       implémente l'interface
                                       CredentialGenerator. Cette classe utilisée pour
obtenir les données d'identification des clients.

  -cgp,--credGenProps <property>         Spécifie les propriétés de la classe
                                       d'implémentation
                                       CredentialGenerator. Les propriétés sont
paramétrées
                                       sur l'objet à l'aide de la méthode
                                       setProperties(String).

  -cxpv,--contextProvider <provider>     Fournisseur de contexte. Exemples : IBMJSSE2,
                                       IBMJSSE, IBMJSSEFIPS.
```

```
  -h,--help <cmdName>                       Appelle l'aide de ligne de commande générale

  -ks,--keyStore <filePath>                 Chemin absolu du fichier de clés. Exemple :
                                           /etc/test/security/server.public

  -ksp,--keyStorePassword <password>    Spécifie le mot de passe d'accès au fichier de
clés.

  -kst,--keyStoreType <type>                Type de fichier de clés. Exemple : JKS, JCEK,
                                           PKCS12.

  -lc,--listCommands <cmdGroupName>     Liste toutes les commandes d'un groupe de
commandes

  -lcg,--listCommandGroups                  Liste tous les groupes de commandes

  -lpc,--listPrivateCommands             Liste toutes les commandes privées.

  -prot,--protocol <protocol>               Protocole. Exemples : SSL, SSLv2,
                                           SSLv3, TLS, TLSv1.

  -pwd,--password <password>                Données d'identification de sécurité de mot de
passe eXtreme
                                           Scale

  -sp,--secProfile <profileName>            Spécifie un nom de profil.

  -ssp,--saveSecProfile <profileName>   Enregistre les valeurs des paramètres de sécurité
dans un
                                           profil de sécurité.

  -to,--timeout <serverTimeout>         Délai de connexion du serveur en secondes.

  -trf,--traceFile <filePath>               Spécifie le chemin absolu du fichier de trace
                                           généré pour la sortie de la commande
                                           xscmd

  -trs,--traceSpec <traceSpec>              Spécifie la spécification de trace de la sortie
de la commande
                                           xscmd

  -ts,--trustStore <filePath>               Chemin absolu du fichier de clés certifiées.
Exemple :
                                           /etc/test/security/server.public

  -tsp,--trustStorePassword <password>   Mot de passe du fichier de clés certifiées

  -tst,--trustStoreType <type>              Type de fichier de clés certifiées. Exemples :
JKS, JCEK,
                                           PKCS12.

  -tt,--transportType <type>                Configuration de type de sécurité de couche de
                                           transport. Exemples : TCP/IP,
                                           SSL-Supported, SSL-Required.

  -user,--username <username>               Données d'identification sécurité de nom
d'utilisateur eXtreme
                                           Scale
```

Toutes les commandes

Voici la liste complète des commandes de l'utilitaire **xscmd**.

Remarque : Les noms de colonne peuvent changer (sauf pour la position et le type d'une colonne) et de nouvelles colonnes peuvent être ajoutées à la fin d'une table si une commande est améliorée de manière appropriée.

Pour obtenir davantage d'aide sur une commande ainsi que la liste de ses paramètres, entrez **xscmd -h nom_commande**. Si vous envisagez de développer un script personnalisé avec ces commandes, vous devez

rediriger le chemin d'erreur standard vers un périphérique NULL en exécutant la commande : `./xscmd.sh -c nomCommande -Options 2> /dev/null`

Exemples

Les exemples suivants montrent comment exécuter une commande à partir de l'utilitaire **xscmd** :

```
./xscmd.sh -lc * Cette commande affiche la liste de toutes les commandes disponibles

./xscmd.sh -h showMapSizes * Cette commande affiche l'aide de la commande showMapSizes
```

Important : La sortie et l'utilisation des commandes suivantes sont susceptibles d'être modifiées dans l'avenir. Si une commande est accompagnée d'un astérisque (*), cela signifie que seule la sortie de la commande est susceptible d'être modifiée :

- listHosts
- showPlacement
- placementServiceStatus
- showDomainReplicationState
- showReplicationState
- *osgiAll
- *osgiCheck
- *osgiCurrent
- *osgiUpdate
- *showLinkedPrimaries
- *triggerPlacement

```
Nom de la commande         Description
------------               -----------
balanceShardTypes      Tentative de redistribution de fragments pour que le taux de
fragments
                           primaires et réplique dans chaque
                           serveur de conteneur corresponde à un fragment.

balanceStatus          Vérifie l'état d'équilibre de la grille de données pour l'élément
                           l'ObjectGrid et le groupe de mappes indiqués.

clearGrid              Efface les données de la grille de données.

dismissLink                Se déconnecte du domaine de service de catalogue
                           spécifié.

enableForPlacement         Réactive le placement des fragments dans un conteneur désactivé
pour le placement
                           des fragments suite à l'échec
                           d'un placement de fragment.

establishLink              Se connecte au domaine de service de catalogue défini
                           avec les noeuds finaux de service de catalogue spécifiés.

findByKey              Recherche les clés correspondantes dans une mappe.

getCatTraceSpec            Extrait la spécification de trace pour tous les
                           services de catalogue connus du processus.

getNotificationFilter      Affiche les filtres de notification pour les serveurs
                           de l'environnement.

getStatsSpec           Extrait la spécification des statistiques.

getTraceSpec           Extrait la spécification de trace.

listAllJMXAddresses    Affiche toutes les adresses de serveur de bean gérés JMX.

listCoreGroups         Liste tous les groupes centraux.

listDisabledForPlacement   Liste des conteneurs qui sont désactivés pour
                           un positionnement de fragment car ce type d'opération
                           échouent.

listHosts                  Liste tous les hôtes. La syntaxe de la commande et la sortie
```

```
sont
                        susceptibles d'être modifiées ultérieurement.

listIndoubts            Liste et résout les transactions en attente de validation.

listObjectGridNames     Liste toutes les instances ObjectGrid et tous les groupes de
mappes connus.

listProfiles            Liste les profils.

listenForNotifications  S'abonne aux notifications reçues par le concentrateur
                        de messagerie. Affiche les erreurs, les avertissements
et les
                        autres messages reçus.

osgiAll                 Affiche tous les classements de services OSGi disponibles. Utilisez
l'option
                            -sn pour afficher un seul service. Seule la sortie de la
commande
                             est susceptible d'être modifiée ultérieurement.

osgiCheck                   Détermine si les classements de service OSGi spécifiés sont
                            disponibles. La sortie de la commande est
                            est susceptible d'être modifiée ultérieurement.

osgiCurrent                 Affiche tous les classements de service OSGi utilisés.
                            Utilisez l'option -sn pour afficher un seul service.
                            La sortie de la commande uniquement est
                            est susceptible d'être modifiée ultérieurement.

osgiUpdate                  Met à jour les services OSGi vers les classements spécifiés
                            La sortie de la commande uniquement est
                            est susceptible d'être modifiée ultérieurement.

overrideQuorum              Indique au serveur de catalogue de remplacer le quorum.

placementServiceStatus Affiche l'état de l'opération de placement ObjectGrid.
                           La syntaxe et la sortie de la commande sont
                           susceptibles d'être modifiées ultérieurement.

releaseShard                Libère le fragment primaire spécifié
                            du serveur de conteneur spécifié.

removeProfile               Retire le profil du système de fichiers.

reserveShard                Réserve le fragment primaire sur le
                            serveur de conteneur spécifié.

resumeBalancing             Tente d'équilibrer et autorise les tentatives futures
d'équilibrage
                            pour l'instance ObjectGrid spécifiée et l'ensemble
                            de mappes définis.

revisions                   Affiche tout l'historique de révision connu.

routetable                  Affiche la table de routage en cours.

setCatTraceSpec             Définit la spécification de trace de tous les serveurs
                            catalogue connus du processus.

setNotificationFilter       Définit le filtre de notification. Le concentrateur de
messagerie
                            traite les messages INFO, WARNING et SEVERE
                            qui correspondent à l'expression régulière.

setStatsSpec                Définit la spécification de statistiques.

setTraceSpec                Spécification de trace dans le format :
```

```
traceType1=traceLevel1=traceState1[:traceTypeN=traceLevelN=traceStateN]*

showCoreGroupMembers       Affiche tous les membres du groupe central.

showDomainReplicationState Affiche les révisions entrantes et sortantes à
                           répliquer entre les fragments primaires dans les domaines
                           liés pour chaque conteneur dans tous les domaines.
                           La syntaxe et la sortie de la commande sont susceptibles
                           d'être modifiées ultérieurement.
showInfo                   Extrait les spécifications d'environnement, y compris
                           les versions installées et les informations relatives à la JVM.

showLinkedDomains          Affiche les domaines externes liés.

showLinkedPrimaries        Affiche les fragments primaires et tous leurs
                           fragments primaires liés locaux et externes.
                           Seule la sortie de la commande est
                           susceptible d'être modifiée ultérieurement.

showMapSizes               Affiche toutes les tailles de mappe.

showNotificationHistory    Affiche les derniers erreurs, avertissements
                           messages stockés dans
                           le concentrateur de messagerie.

showPlacement              Liste tous les serveurs de conteneur et leurs fragments.
                           La syntaxe et la sortie de la commande sont
                           susceptibles d'être modifiées ultérieurement.

showProfile                Affiche le détail du profil spécifié.

showQuorumStatus           Affiche l'état du quorum du serveur de catalogue.

showReplicationState       Affiche les révisions entrantes et sortantes
                           à répliquer entre les fragments primaires et les fragments
                           pour chaque conteneur. La syntaxe de la commande et la sortie
sont
                           susceptibles d'être modifiées ultérieurement.

showTransport              Affiche le transport utilisé par le
                           domaine du service de catalogue. Les types sont notamment ORB
et eXtremeIO.

suspendBalancing           Empêche les tentatives d'équilibrage de l'ObjectGrid et du
groupe de mappes
                           définis.

swapShardWithPrimary       Remplace le fragment réplique défini du conteneur de
                           serveur spécifié
                           par son fragment primaire.

teardown                   Arrête une liste de serveurs de catalogue et de conteneur.

triggerPlacement           Déclenche une opération de placement pour l'instance
                           ObjectGrid et le groupe de mappes. La valeur
                           numInitialContainers est ignorée.
                           Seule la commande est susceptible d'être modifiée
ultérieurement.
```

Rubrique parent : Référence

Tâches associées:
Administration avec l'utilitaire xscmd
Configuration des profils de sécurité pour l'utilitaire xscmd
Demande, affichage et invalidation des données

HTTP command interface reference

With the HTTP command interface, you can run operations on your appliance, configure appliance settings, and administer data grids, collectives, and zones.

Table of contents

List of APPLIANCE commands

CreateAdminTrace	
Description:	Adds or modifies an existing administrative trace string.
Required Parameters:	**stopOnTaskFailure** **trace**

traceName	Specifies the name of the trace.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"AutoCustomLogger", "command": "CreateAdminTrace" } }
```

Command Type:

appliance

CreateAggregateInterface

De

script ion:

Creates a new Aggregate Interface.

Required Parameters:

members

Lists all ethernet interface members of this aggregation.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

primary_member

Specifies which of the aggregated ethernet interfaces is the primary one.

aggregation_policy	Specifies the aggregation policy of the interface.
address	Specifies IP address of the interface.
name	Specifies the interface name.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

{ "task": { "member": "eth0,eth1", "stopOnTaskFailure": "true", "primary_member": "eth0", "aggregation_policy": "active-backup", "address": "1:2:3:4/24", "name": "agg1", "command": "CreateAggregateInterface" } }

Command Type:

appliance

CreateDataCacheTrace

Description:

Adds or modifies an existing data cache trace string.

Required Parameters:

stopOnTask:

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Failure traceName

Specifies the name of the trace.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"autoDataCacheLogger", "command": "CreateDataCacheTrace" } }
```

Command Type

appliance

p
e:

CreateSNMPCommunity

Description:

Creates a Simple Network Management Protocol (SNMP) community.

Required Parameters:

hostRestriction

(Optional) Specifies an IP address on which to restrict SNMP communication. Communication with any other IP address or subnet is denied.

stopOnTaskFailur

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

e

c
o
m
m
u
ni
t
y
N
a
m
e

Specifies the name of the SNMP community to create.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "hostRestriction": "autorestriction", "stopOnTaskFailure":
"true", "command": "CreateSNMPCommunity", "communityName":
"autocommunity" } }
```

Command Type

appliance

p
e:

DeleteSNMPCommunity

D
e
s
cr
ip
ti
o
n:

Deletes a Simple Network Management Protocol (SNMP) community.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
ai
lu
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

c
o
m
m
u
ni
t
y
N
a
m

Specifies the name of SNMP community to delete.

e

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"DeleteSNMPCommunity", "communityName": "autocommunity" } }
```

Command Type:

appliance

EnableCreateAccount

Description:

Enables a setting that allows users to initiate the creation of their

tion: own accounts.

Required Parameters:

stopOnTaskFailure Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

enable Enables account creation. Set the value to false to disable account creation. Set the value to true to enable account creation.

Result Parameter

rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "enable": "true", "command": "EnableCreateAccount" } }
```

Command Type:

appliance

EnablePasswordReset

Description:

Enables the ability to reset the xcadmin password with a serial connection. No other credentials or SMTP messages are required.

Required Parame

Parameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
enable	Enables the password to be reset. Set the value to true to enable password resets.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

E

Example:	{ "task": { "stopOnTaskFailure": "true", "enable": "true", "command": "EnablePasswordReset" } }
Command Type:	appliance

EnableSNMP	
Description:	Enables or disables Simple Network Management Protocol (SNMP).
Required Parameters.	
stopOnTaskFa	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

failure

enable

Enables SNMP. Set the value to false to disable SNMP. Set the value to true to enable SNMP.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "enable": "true", "command":
"EnableSNMP" } }
```

Command Type:

appliance

ModifyAdminDefaultTrace

Description: Changes the trace level for the administrative default logger.

Required Parameters:

stopOnTaskFailure
Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

traceLevel
Specifies the trace level. Valid values: OFF, SEVERE, WARNING, or INFO.

Re

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceLevel": "SEVERE",
"command": "ModifyAdminDefaultTrace" } }
```

Command Type: appliance

ModifyAdminTrace
Description: Changes an administrative logger trace level.

Re

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

traceLevel

Specifies the trace level. Valid values: OFF, SEVERE, WARNING, INFO, FINE, FINER, FINEST, ALL.

traceName

Specifies the name of the trace to modify.

Result Para

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceLevel": "SEVERE",
"traceName": "autoCustomLogger", "command": "ModifyAdminTrace"
} }
```

Command Type:

appliance

ModifyAdministratorEmail

Description:

Changes the default administrator email address.

Required P

Parameters:

StopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
administratorEmail	Specifies a new email address for the default administrator account.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "administratorEmail":
"somenewname@us.ibm.com", "command":
"ModifyAdministratorEmail" } }
```

Command Type:

appliance

ModifyAdministratorName

Description:

Changes the default administrator account name.

Required Parameters:

st

opOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

administratorName

Specifies the new name for the default administrator account.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

E

Example:	{ "task": { "stopOnTaskFailure": "true", "administratorName": "somenewname", "command": "ModifyAdministratorName" } }
Command Type:	appliance

ModifyAdministratorPassword	
Description:	Changes the password for the default administrator account.
Required Parameters.	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

newPassword

Specifies a new default administrator account password.

passwordVerification

Specifies a new default administrator account password, entered a second time for verification.

Response Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "newPassword": "somepass",
"passwordVerification": "somepass", "command":
"ModifyAdministratorPassword" } }
```

Command Type:	appliance

ModifyAggregateInterface	
Description:	Modifies Aggregate Interface parameters.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
n	

name	Specifies the interface name.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:


```
{ "task": { "mtu": "1500", "member": "eth0,eth1",
"stopOnTaskFailure": "true", "transmit_hash_policy": "layer2",
"dad_retransmit_timer": "1000", "use_slaac": "true", "mode": "Auto",
"use_dhcp": "false", "use_arp": "true", "primary_member": "eth0",
"aggregation_policy": "active-backup", "address": "1:2:3:4/24",
"name": "agg1", "command": "ModifyAggregateInterface",
"dad_transmits": "1", "lacp_selection_logic": "stable",
"ipv4_default_gateway": "1:2:3:5" } }
```


Command Type:

appliance

ModifyDataCacheTrace

Des

cr
ip
ti
o
n:

Adds or modifies an existing data cache trace string.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
ai
lu
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

tr
a
c
e
L
e
v
el

Specifies the trace level. Valid values: OFF, SEVERE, WARNING, INFO, FINE, FINER, FINEST, ALL.

tr
a
c
e
N
a
m
e

Specifies the name of the trace.

R

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceLevel": "SEVERE",
"traceName": "autoDataCacheLogger", "command":
"ModifyDataCacheTrace" } }
```

Command Type:

appliance

ModifyEthernetInterface

Description:

Modifies Ethernet Interface parameters.

Result:

equiredParameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
name	Specifies the interface name.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
comma	

ndResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "mtu": "1500", "use_dhcp": "false", "use_arp": "true",
"stopOnTaskFailure": "true", "dad_retransmit_timer": "1000",
"use_slaac": "true", "address": "1:2:3:4/24", "name": "eth0",
"dad_transmits": "1", "command": "ModifyEthernetInterface",
"ipv4_default_gateway": "1:2:3:5", "mode": "Auto" } }
```

Command Type: appliance

ModifyLDAP

Description:

Configures the appliance to use Lightweight Directory Access Protocol (LDAP) for user login authentication.

Required Parameters:

jndiSecurityPri	Specifies the JNDI security principal.

nci pal	
sto pOn Tas kFai lure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
lda pUs erId Sea rch Filt erP atte rn	LDAP user search filter pattern.
ne wPa ssw ord	Specifies the new LDAP password.
gro upB ase Dn	Specifies the LDAP JNDI base distinguished name (DN) for groups.
pas swo rdV erifi cati on	Specifies the new LDAP password a second time for password verification. The value must be the same as the newPassword parameter.
use rBa seD n	Specifies the LDAP JNDI base distinguished name (DN) for users.
ena ble	Specifies if LDAP user authentication is enabled. Set the value to true to enable LDAP user authentication.
jndi Pro vid erU RL	Specifies the URL for the LDAP provider.

Result Parameter

rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

E
x
a
m
pl
e:

```
{ "task": { "stopOnTaskFailure": "true", "jndiSecurityPrincipal":
"CN=Administrator,CN=users,DC=mycompany,DC=com",
"ldapUserIdSearchFilterPattern": "thefilterpattern", "newPassword":
"somepassword", "passwordVerification": "somepassword",
"groupBaseDn": "cn=group", "userBaseDn": "cn=user", "enable":
"true", "command": "ModifyLDAP", "jndiProviderURL": "someurl" } }
```

C
o
m
m
a
n
d
T
y
p
e:

appliance

ModifyNtpServers

D
e
s
cr
ip
ti
o
n:

Changes the list of Network Time Protocol (NTP) servers.

R
e
q
ui
r
e
d
P
a
r
a
m

eters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

listOfServers

Specifies a list of comma-separated Network Time Protocol (NTP) servers.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.

commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "listOfServers":
"time.nist.gov,10.10.2.2", "command": "ModifyNtpServers" } }
```

Command Type:

appliance

ModifySMTPEmail

Description:

Changes the reply-to email address that is used for sending passwords that are reset by users.

Required Parameters:

stopOn	Specifies whether to stop running the batch routine

TaskFailure	when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
emailAddress	Specifies the reply-to email address. Use the email address for the administrator.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ModifySMTPEmail", "emailAddress": "admin@mycompany.com" } }
```

Co

Command Type:	appliance

ModifySMTPServer

Description:	Changes the Simple Mail Transfer Protocol (SMTP) server.
Required Parameters:	**stopOnTaskFailure** Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only. **host**

Name	Specifies the SMTP host address.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ModifySMTPServer", "hostName": "sicdsjc" } }
```

Command Type:

appliance

ModifySearchFilterUsers

De

Description:

Changes the Lightweight Directory Access Protocol (LDAP) search filter for users.

Required Parameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
ldapUserIdSearchFilterPattern	LDAP user search filter pattern.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true",
"ldapUserIdSearchFilterPattern": "(&(uid={0})
(objectclass=inetOrgPerson))", "command":
"ModifySearchFilterUsers" } }
```

Command Type: appliance

ModifyTimeZone

Description: Changes the time zone for the appliance.

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

localTimeZone

Specifies the new time zone as an abbreviated string. Valid values: EST : Eastern Standard Time CST : Central Standard Time MST : Mountain Standard Time PST : Pacific Standard Time HAST : Hawaii AKST : Alaska AST : Atlantic UTC : Coordinated Universal Time GMT : Greenwich Mean Time CET : Central Europe EET : Eastern Europe MSK : Moscow AST_ARABIA : Arabia KRT : Pakistan IST : India NOVT : Novosibirsk CST_CHINA : China JST : Japan AWST : Australia West ACST : Australia Central AEST : Australia East

Result Parameters:

errorMessage A message that explains the reason for the failure, if the command failed.

status Specifies the status of the command that was run. The result can be either success or failed.

commandResult A JSON-formatted statement that contains the result of the command that was run.

commandName Specifies the name of the HTTP command interface command that was run.

Example

{ "task": { "stopOnTaskFailure": "true", "command":

ple:	"ModifyTimeZone", "localTimeZone": "EST" } }
Command Type:	appliance

PingRemoteHost	
Description:	Tests the visibility of a host from this appliance.
Required Parameters:	
stopOnTaskFailur	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

e

hostName

Specifies the IP address or host name of the remote host.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"PingRemoteHost", "hostName": "io03.rtp.raleigh.ibm.com" } }
```

Command Type:

appliance

RemoveAdminTrace

Description:

Removes an existing administrative console trace string.

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

traceName

Specifies the name of the trace.

Result

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"AutoCustomLogger", "command": "RemoveAdminTrace" } }
```

Command Type: appliance

RemoveAggregateInterface

Description: Deletes Aggregate Interface of a given name.

Requi

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

name

Specifies the interface name.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "name": "agg1", "command": "RemoveAggregateInterface" } }
```

Command Type:

appliance

RemoveDataCacheTrace

Description:

Removes an existing data cache trace string.

Required Parameters:

stopOnT	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the

taskFailure	batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
traceName	Specifies the name of the trace.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"autoDataCacheLogger", "command": "RemoveDataCacheTrace" } }
```

Command

appliance

Type:

RestartAppliance

Description:

Restarts or shuts down the appliance.

Required Parameters:

immediate

Specifies if the appliance waits to complete active tasks before restarting. If the value is set to true, the appliance restarts immediately. If the value is set to false, the appliance waits for all active tasks to complete before restarting.

stopOnTaskFailur

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

e **s** **h** **u** **t** **d** **o** **w** **n**	Specifies if the appliance restarts or shuts down. If the value is set to true, the appliance is shut down. If the value is set to false, the appliance restarts.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "immediate": "false", "stopOnTaskFailure": "true",
"command": "RestartAppliance", "shutdown": "true" } }
```

Command Type:

appliance

ViewAllAggregateInterfaces	
Description:	Displays the information for every aggregate interface.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
Result Parameter	

e
rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

E
x
a
m
pl
e:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ViewAllAggregateInterfaces" } }
```

C
o
m
m
a
n
d
T
y
p
e:

appliance

ViewAllEthernetInterfaces

D
e
s
cr
ip
ti
o
n:

Displays the information for every Ethernet Interface.

R
e
q
ui
r
e
d
P
a
r
a

Parameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ViewAllEthernetInterfaces" } }
```

Command Type: appliance

List of COLLECTIVE commands

AddApplianceToCollective

Description: Adds an existing appliance to the current collective.

Required Parameters:

appliancelPos

Specifies the IP address of the appliance to add to the collective.

nTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
secretKey	Specifies the secret key of the appliance.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
zoneName	Specifies the name of the zone to which you want to assign the appliance.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "applianceIP": "192.168.222.128", "stopOnTaskFailure":
"true", "secretKey": "P5Xa4F8MQeSxuuhKxnaStw==", "waitOnTask":
"true", "command": "AddApplianceToCollective", "zoneName":
"DefaultZone" } }
```

Command Type:

collective

AddRoleToGroup

Description:

Assigns an access role to a group. The defined roles are: appliance administration, appliance monitoring, or data cache creation.

Required Parameters:

g

groupName

Specifies the name of the group that is being assigned to a role.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

role

Specifies the role to assign to the group. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Ex

Example:	{ "task": { "stopOnTaskFailure": "true", "groupName": "autoGroup1003", "command": "AddRoleToGroup", "role": "5" } }
Command Type:	collective

AddRoleToUser

Description:	Assigns an access role to a user. The defined roles are: appliance administration, appliance monitoring, or data cache creation.
Required Parameters:	
stopOnTaskFailu	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

role

Specifies the role to assign to the user. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

userName

Specifies the name of the user that is being assigned a role.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "AddRoleToUser",
"role": "2", "userName": "autoUser1003" } }
```

CommandT

collective

y
p
e:

AddUserToGroup

D
e
s
cr
ip
ti
o
n:

Adds a user to a group. After the user is added to the group, the user has the roles that are assigned to the selected group only. Any individual user roles are lost.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

g r o u p N a m e

Specifies the name of a defined group.

s t o p O n T a s k F ai lu r e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

userName	Specifies the name of a defined user.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "AddUserToGroup", "userName":
"autoUser1003" } }
```

Command Type:

collective

CreateCollectiveLink

Description:

Creates a link to another collective.

Required Parameters:

appliance Name

Specifies the host name or IP address of an appliance that is a member of another collective.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

remotePassword

Specifies the password that is used to log in to the remote appliance.

remoteUserName

Specifies the user name that is used to log in to the remote appliance.

Response Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Ex

```
{ "task": { "applianceName": "myappliance.mydomain.com",
```

Example:

"stopOnTaskFailure": "true", "command": "CreateCollectiveLink", "remotePassword": "somePassword", "remoteUserName": "someUser" } }

Command Type:

collective

CreateGroup

Description:

Creates a new group. Groups are useful for assigning multiple users the same set of roles.

Required Parameters:

groupName

st

Specifies a name for the new group. Group names must be unique.

opOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

groupDescription

Specifies a description of the group and its users.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Ex

Example:	{ "task": { "stopOnTaskFailure": "true", "groupName": "autoGroup1003", "groupDescription": "this group is huge", "command": "CreateGroup" } }
Command Type:	collective

CreateUser

Description:	Creates a new user.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
stopOnTaskFailurep	

passwordVerification	Specifies the password, entered a second time for verification.
userPassword	Specifies a new password for the selected user.
emailAddress	Specifies an email address for the new user.
fullUserName	Specifies a full name for the user. Example: John E. Doe.
userName	Specifies a short name for the new user. Example: xcadmin.

ResultParameter

rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "passwordVerification":
"someuserpass", "userPassword": "someuserpass", "command":
"CreateUser", "emailAddress": "someuser@us.ibm.com",
"fullUserName": "someuser isme", "userName": "someuser" } }
```

Command Type:

collective

DeleteCollectiveLink

Description:

Removes the link between the local collective and the specified collective.

Required Parame

Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

name

Specifies the name of the collective to delete.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example

```
{ "task": { "stopOnTaskFailure": "true", "name": "myCollectiveName",
"command": "DeleteCollectiveLink" } }
```

Command Type:	collective

DeleteGroup

Description:	Deletes a group. Users that belonged to the group no longer belong to the group and might lose any assigned roles.
Required Parameters:	
groupName	Specifies the name of a defined group.
stopOn	

TaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "DeleteGroup" } }
```

Command Type:

collective

DeleteUser

Description:

Deletes a user. If the user belongs to a group, the user is removed from the group.

Required Parameters:

StopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

UserName

Specifies the name of a defined user.

Result

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "DeleteUser",
"userName": "autoUser1003" } }
```

Command Type:

collective

GetCollectiveName

Description:

Retrieves the collective name.

Require

edParameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Ex

Example:	{ "task": { "stopOnTaskFailure": "true", "command": "GetCollectiveName" } }
Command Type:	collective

GetHealthStatus	
Description:	Displays the hardware and software Health Status information for the appliance.
Required Parameters:	
stopOnTaskFa	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

il
u
r
e

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "GetHealthStatus" } }
```

Command Type:

collective

ModifyGroupDescription

De

scription:

Changes the description of a selected group.

Required Parameters:

groupName

Specifies the name of a defined group.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

groupDesc

Specifies a description of the group and its users.

ri
p
ti
o
n

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName": "somegroup",
"groupDescription": "somegroupdesc", "command":
"ModifyGroupDescription" } }
```

Command Type:

collective

ModifyScheduleExportSettings

De

script ion:

Enable and disable the scheduled exports feature. Also, specify an interval to generate a configuration export.

Required Parameters:

daylinterval

Specifies how many days to wait in between scheduled configuration exports.

time

Specifies the scheduled date that the first export will occur. Must be in the following format: YYYY-MM-DD. For example, "2014-09-08".

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

enable

Specifies if the scheduled configuration export feature is enabled or disabled.

date	CLI.parameter.date.desc

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "dayInterval": "3", "stopOnTaskFailure": "true", "time":
"16:25", "enable": "true", "command":
"ModifyScheduleExportSettings", "date": "2014-02-14" } }
```

Command Type:

collective

ModifyTransportAndStorageMode

De

script:

Changes the transport and storage mode setting for the collective. When you update the transport and storage mode, a task is created to update the collective settings and all appliances in the collective are restarted.

Required Parameters:

transport

Specifies the transport and storage mode. Possible values are ORB, XIO, XIO_XM. Transport and storage modes enable the exchange of objects and data between different server processes. When you enable IBM eXtremeIO (XIO), relative response time is faster than the Object Request Broker (ORB). ORB is deprecated. With XIO enabled, you can also create an enterprise data grid. With an enterprise data grid, client applications that are written in different programming languages, such as Java and .NET, can access the same data grid. When you use IBM eXtremeMemory, cache entries are stored in native memory. When you use heap memory, cache entries are stored in the Java heap.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "command": "ModifyTransportAndStorageMode", "mode":
"XIO" } }
```

Command Type: collective

ModifyUserEmail

Description: Changes the email address for a user.

Required Parameters:

st

opOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

emailAddress

Specifies a new email address for the user.

userName

Specifies the user name of the user to update.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandRes

A JSON-formatted statement that contains the

ult	result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ModifyUserEmail", "emailAddress": "user1@mycompany.com",
"userName": "user1" } }
```

Command Type: collective

ModifyUserPassword

Description: Changes the password for a specified user.

Required Parameters:

stopOn	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the

TaskFailurenewPassswordpasswordVerificationuserName	batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only. Specifies a new password for the selected user. Specifies the new password for the selected user, entered a second time for verification. Specifies the name of the user to update.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
comma	

ndResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "newPassword": "newpass",
"passwordVerification": "newpass", "command":
"ModifyUserPassword", "userName": "autoUser1003" } }
```

Command Type: collective

RemoveApplianceFromCollective

Description: Removes an existing appliance from the current collective.

Required Parameters:

appli

a
n
c
e
l
P

Specifies the IP address of the appliance to be removed.

s
t
o
p
O
n
T
a
s
k
F
a
il
u
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

w
a
it
O
n
T
a
s
k

Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.

R
e
s
ul
t
P
a
r
a
m
e
t
rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
comma	

ndName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "applianceIP": "192.168.222.128", "stopOnTaskFailure":
"true", "waitOnTask": "true", "command":
"RemoveApplianceFromCollective" } }
```

Command Type:

collective

RemoveRoleFromGroup

Description:

Removes an access role from a group. The defined roles are: appliance administration, appliance monitoring, or data cache creation.

Required Parameters:

groupNa	Specifies the name of the group for which you are removing an access role.

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
role	Specifies the role to remove from the group. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "RemoveRoleFromGroup", "role": "5"
} }
```

Command Type:	collective

RemoveRoleFromUser

Description:	Removes an access role from a user. The defined roles are: appliance administration, appliance monitoring, or data cache creation.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
role	Specifies the role to remove from the user. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

username

Specifies the name of for which you are removing an access role.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"RemoveRoleFromUser", "role": "2", "userName": "autoUser1003" } }
```

Command Type:

collective

RemoveUserFromGroup

Description: Removes a user from a group. After a user is removed from a group, the user continues to have the same roles as the group. However, the user roles for the user no longer change when the group roles change. You must modify the user roles for the selected user individually.

Required Parameters:

groupName Specifies the name of a defined group.

stopOnTaskFailure Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

userName Specifies the name of a defined user.

a
m
e

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "RemoveUserFromGroup",
"userName": "autoUser1003" } }
```

Command Type:

collective

ViewAllGroups

Descr

ip
ti
o
n:

Displays information about all groups.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
a
il
u
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

R
e
s
ul
t
P
a
r
a
m
e
t
e
rs
:

errorM
essage

A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewAllGroups"
} }
```

Command Type: collective

ViewAllUsers

Description: Displays the information for every user.

Required Parameters:

s

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewAllUsers" }
}
```

Command Type

collective

p
e:

ViewCollectiveLinks

D
e
s
cr
ip
ti
o
n:

Displays the information for every collective link.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
a
il
u
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

R
e
s
ul
t
P
a
r
a

meters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewCollectiveLinks" } }
```

Command Type:

collective

ViewGroup

Description:

Displays information about a selected group.

Required Pa

Parameters:

groupName	Specifies the name of a defined group.
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

ResultParameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.

commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "ViewGroup" } }
```

Command Type: collective

ViewMemberDetails

Description: Displays information about a member in a collective.

Required Parameters:

appliance	Specifies the IP Address of the member.

cIP stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "applianceIP": "192.168.222.128", "stopOnTaskFailure":
"true", "command": "ViewMemberDetails" } }
```

Co

Command Type:	collective

ViewTransportAndStorageMode	
Description:	Displays the transport and storage mode.
Required Parameters:	
Result Parameters:	

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "command": "ViewTransportAndStorageMode" } }
```

Command Type:

collective

ViewUser

Description:

Displays information about a selected user.

Required Parameters:

s

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

userName

Specifies the name of a defined user.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewUser",
"userName": "autoUser1003" } }
```

Command Type: collective

List of GRID commands

ClearGrid	
Description:	Clears data from a data grid.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to be cleared.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "ClearGrid" } }
```

Command Type:

grid

CreateGrid

Description: Creates a new simple, dynamic cache, or session data grid.

Required Parameters:

stopOnTaskFailure — Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName — Specifies the name of the data grid to create.

waitOnTas — Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.

k

g
ri
d
T
y
p
e

Specifies the type of data grid to create. Valid values: simple, dynamic, or session.

Response Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"waitOnTask": "true", "command": "CreateGrid", "gridType": "simple"
} }
```

Command Type:

grid

DeleteGrid

Description:

Deletes a data grid.

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to delete.

waitOnTask

Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.

k

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "DeleteGrid" } }
```

Command Type:

grid

ExportGrid

Description:

(Simple data grids only) Exports data grid information to XML.

on:

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to export.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "ExportGrid" } }
```

Command Type:

grid

GrantGroupAccess

Description:

Gives a group access rights to a data grid.

Required Parameter

rs
:

groupName

Specifies the name of the group to grant the given access rights.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to grant access rights.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

comma

A JSON-formatted statement that contains the

ndResult	result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName": "somegroup",
"gridName": "myGrid", "command": "GrantGroupAccess" } }
```

Command Type: grid

GrantUserAccess

Description: Gives a user access rights to a data grid.

Required Parameters:

stop

OnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to which the user is assigned access.

userName

Specifies the name of the user to give access rights.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

E

Example:	{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid", "command": "GrantUserAccess", "userName": "someuser" } }
Command Type:	grid

ModifyGridCapacity	
Description:	Changes the capacity for a simple data grid.
Required Parameters.	
stopOnTaskF	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

failureUseLRUgridName	(Simple data grids only) Enables least recently used (LRU) eviction when set to true.
waitOnTaskgridCapLimit	Specifies the name of the data grid to update.
	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
	Specifies the maximum capacity for the selected data grid in megabytes.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "useLRU": "false",
"gridName": "myGrid", "waitOnTask": "true", "command":
"ModifyGridCapacity", "gridCapLimit": "10" } }
```

Command Type:

grid

ModifyGridReplication

Description:

Changes the replication settings for a data grid.

Required Parameters:

st

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
maximumSyncReplicas	Specifies the maximum number of synchronous replicas for this data grid.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
maximumAsyncReplicas	Specifies the maximum number of asynchronous replicas for this data grid.

Results Parameters:

errorM

essage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"waitOnTask": "true", "maximumSyncReplicas": "4", "command":
"ModifyGridReplication", "maximumAsyncReplicas": "2" } }
```

Command Type:

grid

ModifyGridSecurity

Description:

Changes the security settings for a data grid.

Required Parameter

rs
:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
authorizationEnabled	Enables authorization for the data grid when set to true.
gridName	Specifies the name of the data grid to update.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
securityEnabled	Enables security for the data grid when set to true.

Result

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "authorizationEnabled":
"false", "gridName": "myGrid", "waitOnTask": "true", "command":
"ModifyGridSecurity", "securityEnabled": "false" } }
```

Command Type:

grid

ModifyGridTTL

Description:

(Simple data grids only) Changes time to live (TTL) settings for the data grid.

Requi

Required Parameters:

defaultMapEvictorType	(Optional) Specifies the time to live eviction type used for the default map. Valid values: NONE, CREATION_TIME, LAST_ACCESS_TIME, LAST_UPDATE_TIME.
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
gridTTL	Specifies the amount of time, in seconds, to keep data before evicting the data from data grid.

Result Para

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "defaultMapEvictorType": "CREATION_TIME",
"stopOnTaskFailure": "true", "gridName": "myGrid", "waitOnTask":
"true", "command": "ModifyGridTTL", "gridTTL": "60000" } }
```

Command Type:

grid

ModifyGroupAccess

Description:

Changes the group access rights for a data grid.

Required P

Parameters:

groupName	Specifies the name of the group for which you want to modify access rights.
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
accessType	Specifies the type of access to give to the group. Valid values: 1 = read, 2 = write, 3 = create, 4 = all.

Resul

tParameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName": "somegroup",
"gridName": "myGrid", "command": "ModifyGroupAccess",
"accessType": "1" } }
```

CommandType:

grid

ModifyUserAccess

Description:

Changes user access rights for a data grid.

Req

required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
userName	Specifies the name of the user for which access rights are being updated.
accessType	Specifies the type of access to give to the user. Valid values: 1 = read, 2 = write, 3 = create, 4 = all.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "ModifyUserAccess", "userName": "someuser",
"accessType": "1" } }
```

Command

Fichier de propriétés du client

Création d'un fichier de propriétés en fonction de vos exigences pour les processus client WebSphere eXtreme Scale.

Exemples de fichier de propriétés du client

Java Vous pouvez utiliser le fichier `sampleClient.properties` qui se trouve dans le répertoire *rép_base_wxs*`/properties` pour créer votre propre fichier de propriétés.

.NET Vous pouvez utiliser le fichier `Client.Net.properties` qui se trouve dans le répertoire *net_client_home*`\config\` pour créer votre propre fichier de propriétés.

Java

Définition d'un fichier de propriétés pour les clients Java

Vous pouvez spécifier le fichier de propriétés du client de l'une des manières ci-après. La spécification d'un paramètre en utilisant l'un des éléments plus loin dans la liste remplace le paramètre précédent. Par exemple, si vous spécifiez une valeur de propriété système pour le fichier de propriétés du serveur, les propriétés contenues dans ce fichier remplacent celles contenues dans le fichier `objectGridClient.properties` qui se trouvent dans le chemin d'accès aux classes.

1. En tant que fichier nommé n'importe où dans le chemin d'accès aux classes. Vous ne pouvez pas placer ce fichier dans le répertoire de travail du système :

   ```
   objectGridClient.properties
   ```

2. En tant que propriété système dans une configuration autonome ou WebSphere Application Server. Cette valeur peut spécifier un fichier du répertoire de travail du système, mais pas un fichier figurant dans le chemin d'accès aux classes :

   ```
   -Dobjectgrid.client.props=nom_fichier
   ```

   Remarque : Dans une configuration WebSphere Application Server, le fichier de propriétés du client doit se trouver dans le chemin d'accès aux classes si vous voulez pouvoir imposer l'utilisation de ce fichier à l'aide de la propriété système ; par exemple, *racine_was*`/properties` ou *racine_profil*`/properties`, selon que vous voulez que le fichier de propriétés s'applique à un profil spécifique ou à toute l'installation.

3. Comme une substitution à l'aide d'un programme, à l'aide de la méthode ClientClusterContext.getClientProperties. Les données de l'objet sont alimentées avec celles des fichiers de propriétés. Vous ne pouvez pas configurer les propriétés de sécurité à l'aide de cette méthode.

.NET

Définition d'un fichier de propriétés pour WebSphere eXtreme Scale Client for .NET

Le fichier par défaut des propriétés de client se trouve dans le répertoire *net_client_home*`\config\Client.Net.properties`. Si vous voulez définir vos propres fichiers de propriétés, fournissez le chemin d'accès complet au fichier de propriétés souhaité dans chaque appel de méthode Connect.

```
IGridManager gm = GridManagerFactory.GetGridManager( );
ICatalogDomainInfo cdi = gm.CatalogDomainManager.CreateCatalogDomainInfo( "localhost:2809"
);
ccc = gm.Connect( cdi, @"C:\MyLocation\MyClient.properties" );
grid = gm.GetGrid( ccc, gridName );
```

Propriétés du client

Propriétés du client

.NET 2.5+ **enableDynamicConfiguration**

Lorsque cette propriété a la valeur `true`, les modifications apportées à la propriété requestRetryTimeout du fichier de propriétés du client sont détectées dynamiquement. La nouvelle valeur de cette propriété est immédiatement utilisée pour calculer le nouveau délai d'attente pour les nouvelles tentatives d'exécution de requête.

Valeur par défaut : `false`

Java listenerHost

Indique le nom d'hôte auquel le protocole de transport ORB (Object Request Broker) ou eXtremeIO (XIO) se lie pour les communications. La valeur doit être un nom qualifié complet de domaine ou une adresse IP. Si la configuration implique plusieurs cartes réseau, configurez l'hôte du programme d'écoute et le port d'écoute pour que le mécanisme de transport dans la machine JVM connaisse l'adresse IP de liaison. Si vous ne définissez pas l'adresse IP à utiliser, des symptômes (dépassements du délai de connexion, défaillances inhabituelles d'API et clients qui semblent se bloquer) apparaissent.

Java listenerPort

Indique le numéro de port auquel le protocole de transport ORB (Object Request Broker) ou eXtremeIO (XIO) se lie pour les communications. Le numéro de port défini pour listenerPort est utilisé pour les communications entre un client et un serveur de catalogue, et entre un serveur de conteneur et un serveur de catalogue qui se trouvent dans le même domaine. Il est également utilisé pour les communications interdomaine et intradomaine entre les serveurs de catalogue.

Valeur par défaut :`2809`

Remarque : Lorsqu'un serveur de grille de données s'exécute dans WebSphere Application Server et que le protocole de transport ORB est utilisé, un autre port ORB_LISTENER_ADDRESS doit également être ouvert. Le port BOOTSTRAP_ADDRESS réachemine les requêtes vers ce port. Si vous utilisez le protocole de transport XIO, le port XIO_ADDRESS doit être ouvert.

Java preferLocalProcess

Cette propriété n'est pas utilisée actuellement. Elle est réservée à une utilisation future.

Java preferLocalHost

Cette propriété n'est pas utilisée actuellement. Elle est réservée à une utilisation future.

Java .NET preferZones

Indique une liste de zones de routage recommandées. Chaque zone spécifiée doit être séparée de la précédente par une virgule, selon le format suivant : `preferZones=ZoneA,ZoneB,ZoneC`

Valeur par défaut : aucune

Java .NET requestRetryTimeout

Indique pendant combien de temps (en millisecondes) le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Utilisez l'une des valeurs admises suivantes :

- Une valeur égale à `0` indique que la requête doit échouer immédiatement et ignorer la logique interne régissant les nouvelles tentatives.
- Une valeur égale à `-1` indique que le délai entre les tentatives d'exécution de la requête n'est pas défini, ce qui signifie que la durée pendant laquelle le système tente d'exécuter la requête dépend du délai d'expiration des transactions. (Valeur par défaut)
- Une valeur supérieure à `0` indique la valeur du délai d'expiration de la requête en millisecondes. Les exceptions dont la création échoue sont renvoyées. Même lorsque des exceptions, telles que DuplicateException, sont réexécutées, elles sont également renvoyées quand elles échouent. Le délai de transaction reste utilisé comme délai d'attente maximal.

Java .NET shuffleBootstrapAddresses

Indique si les adresses de service de catalogue doivent subir un traitement aléatoire lorsqu'elles sont utilisées par un client qui s'amorce dans la grille de données. La valeur doit être `true` ou `false`.

Valeur par défaut : `true`

Java 2.5+ xioTimeout

Indique le délai d'attente maximum (en secondes) pour les tentatives d'établissement d'une connexion socket sortante par eXtremeIO. Cette valeur est également utilisée comme délai d'attente par défaut par la logique eXtremeIO interne.

Java 2.5+ xioRequestTimeout

Indique pendant combien de secondes une requête attend une réponse avant d'expirer. Cette propriété influence la durée de la reprise en ligne du client en cas d'indisponibilité du réseau. Si vous spécifiez une valeur trop faible pour cette propriété, les demandes risquent d'arriver à expiration trop tôt. Définissez soigneusement la valeur de cette propriété pour éviter les dépassements de délai d'attente inutiles.

Propriétés de sécurité du client

Propriétés de sécurité générales

Java .NET securityEnabled

Active la sécurité du client WebSphere eXtreme Scale. Ce paramètre doit correspondre au paramètre securityEnabled dans le fichier de propriétés de serveur WebSphere eXtreme Scale. Si les paramètres ne correspondent pas, la connexion à la grille de données échoue.

Valeur par défaut : `false`

Propriétés de configuration de l'authentification des données d'identification

Java .NET credentialAuthentication

Indique la prise en charge de l'authentification des données d'identification du client. Utilisez l'une des valeurs admises suivantes :

- `Jamais` : Le client ne prend pas en charge l'authentification des données d'identification.
- `Pris en charge` : Le client prend en charge l'authentification des données d'identification s'il en est de même pour le serveur. (Valeur par défaut)
- `Obligatoire` : Le client requiert l'authentification des données d'identification.

Java .NET authenticationRetryCount

Indique le nombre de tentatives d'authentification si les données d'identification ont expiré. Si la valeur est 0, aucune nouvelle tentative d'authentification n'est exécutée.

Valeur par défaut : 3

.NET credentialGeneratorAssembly

Indique le nom de l'assemblage utilisé pour générer les données d'identification pour le WebSphere eXtreme Scale Client for .NET. Pour que vous puissiez définir cette propriété, la propriété credentialAuthentication doit avoir la valeur `Pris en charge` ou `Obligatoire`. La valeur indiquée doit être un nom d'assemblage `.dll` C# valide avec version, culture et autres propriétés.

Exemple : `IBM.WebSphere.Caching.CredentialGenerator, Version=8.6.0.0, Culture=neutral, PublicKeyToken=b439a24ee43b0816`

Java .NET credentialGeneratorClass

Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Pour que vous puissiez définir cette propriété, la propriété credentialAuthentication doit avoir la valeur `Pris en charge` ou `Obligatoire`. Cette classe est utilisée pour obtenir les données d'identification des clients.

Valeur par défaut : aucune

Java .NET credentialGeneratorProps

Spécifie les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés sont affectées à l'objet à l'aide de la méthode setProperties(String). Pour que vous puissiez définir cette propriété, la propriété credentialAuthentication doit avoir la valeur `Pris en charge` ou `Obligatoire`. La valeur credentialGeneratorprops n'est utilisée que si la valeur de la propriété credentialGeneratorClass n'est pas NULL.

Propriétés de configuration de la sécurité de la couche de transport

Java .NET transportType

Indique le type de transport du client. Les valeurs possibles sont :

- `TCP/IP` : Indique que le client ne prend en charge que les connexions TCP/IP.
- `SSL pris en charge` : Indique que le client prend en charge les connexions TCP/IP et SSL (Secure Sockets Layer). (Valeur par défaut)
- `SSL requis` : Indique que le client requiert des connexions SSL.

Propriétés de configuration SSL

Java alias

Indique le nom d'alias dans le fichier de clés. Cette propriété est utilisée si le fichier de clés contient plusieurs certificats de paire de clés et que vous souhaitez sélectionner l'un des certificats.

Valeur par défaut : aucune

Java contextProvider

Indique le nom du fournisseur de contexte du service sécurisé. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que le type du fournisseur de contexte est incorrect.

Les valeurs valides sont : `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

Java **keyStore**

Indique un chemin complet vers le fichier de clés.

Exemple :

```
etc/test/security/client.private
```

Java **keyStoreType**

Indique le type de fichier de clés. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : JKS, JCEK, PKCS12, etc.

Java .NET **protocol**

Indique le type du protocole de sécurité à utiliser pour le client. Définissez cette valeur de protocole en fonction du fournisseur de sécurité. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que la valeur du protocole est incorrecte.

Java Les valeurs valides sont : SSL, SSLv3, TLS, TLSv1, etc.

.NET Les valeurs valides sont : SSLv2, SSLv3, TLS ou `Par défaut` (SSLv3 ou TLS1.0)

.NET **publicKeyFile**

Définit le nom de chemin complet d'un fichier qui contient la clé publique exportée depuis le serveur.

Exemple : `c:\tmp\wxs\serverA.cer`

Java **trustStoreType**

Indique le type de fichier de clés certifiées. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : JKS, JCEK, PKCS12, etc.

Java **trustStore**

Indique un chemin complet vers le fichier de clés.

Exemple :

```
etc/test/security/server.public
```

Java **keyStorePassword**

Indique le mot de passe de chaîne du fichier de clés. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

Java **trustStorePassword**

Indique un mot de passe de chaîne au fichier de clés certifiées. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

Rubrique parent : Référence

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de la sécurité du client
.NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Information associée:
Interface ICredential
Interface ICredentialGenerator

Syntaxe d'expression régulière

Une expression régulière est une chaîne structurée utilisée pour la mise en correspondances d'autres chaînes. Vous pouvez utiliser des expressions régulières pour les données d'invalidation et pour le filtrage des messages qui s'affichent dans Message Center.

Exemples d'expression régulière

Expression régulière	Description
`.`	Recherche n'importe quel caractère.
`.*`	Recherche n'importe quel caractère, zéro ou plusieurs fois.
`A.*`	Recherche toutes les chaînes qui commencent par "A".
`.*A`	Recherche toutes les chaînes qui se terminent par "A".
`A?`	Recherche "A", zéro ou une fois.
`A*`	Recherche "A", zéro ou plusieurs fois.
`A+`	Recherche "A", une ou plusieurs fois.
`A?B*C+`	Recherche n'importe quel caractère, suivi de zéro ou un A, suivi de zéro ou plusieurs B, suivi d'un ou plusieurs C
`[ABC].*`	Recherche les chaînes commençant par les lettres "A", "B" ou "C".
`.*ABC.*`	Recherche les chaînes comportant la chaîne "ABC" dans la chaîne.
`(ABC|DEF).*`	Recherche les chaînes commençant par la chaîne "ABC" ou "DEF".

Pour la syntaxe complète d'une expression régulière, consultez la documentation de l'API java.util.regex.Pattern.

Rubrique parent : Référence

Options de configuration de mappe dynamique

Vous pouvez créer des mappes supplémentaires dans une grille de données en demandant à votre application client de se connecter à la mappe spécifiquement nommée. Une fois cette connexion établie, la mappe est automatiquement créée.

Attribution de nom à une mappe dynamique

Lorsque vous créez une grille de données, deux mappes sont créées par défaut. La première prend le nom de la grille de données. Par exemple, si vous créez la grille de données `myGrid`, vous obtenez automatiquement une mappe `myGrid`. La seconde est destinée au cache local et prend le nom `myGrid.NCI`. Cependant, vous pouvez également ajouter des mappes à la grille de données. Une mappe est automatiquement créée lorsque l'application client se connecte à une mappe à l'aide de la convention d'attribution de nom suivante :

```
<nom_mappe>.<modèle>.<option_verrouillage>.<invalidation_cache_local>
```

Où :

***nom_mappe* (obligatoire)**

Indique le nom de la mappe.

***modèle* (obligatoire)**

Indique le modèle qui définit la date d'expiration des entrées de la mappe, en définissant le comportement TTL (Time To Live). Pour obtenir la liste des options disponibles, voir Modèles de mappes.

option_verrouillage

Spécifie le mécanisme de verrouillage utilisé pour la mappe. Pour obtenir la liste des options disponibles, voir Options de verrouillage.

2.5+ ***invalidation_cache_local***

Spécifie si l'invalidation de cache local est utilisé pour la mappe. L'invalidation de cache local est utilisée uniquement lorsque la mappe comporte un type de verrouillage défini sur `NONE` ou `OPTIMISTIC`. Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée sur la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Voir Options d'invalidation du cache local pour obtenir la liste options disponibles.

Vous devez inclure un nom de modèle pour la mappe. Si vous indiquez pas d'option de verrouillage, aucun verrouillage n'a lieu sur la mappe.

Modèles de mappes

Tableau 1. Modèles de mappes dynamiques

Modèle de mappe	Description
*.NONE	Spécifie une mappe sans expiration de durée de vie.
*.LUT	Spécifie une mappe dans laquelle les entrées sont arrivées à expiration sur la base de la date/heure de dernière mise à jour de l'entrée. La durée de vie par défaut est d'une heure.
*.LAT	Une mappe dans laquelle les entrées sont arrivées à expiration sur la base de la date/heure de dernier accès à l'entrée. La durée de vie par défaut est d'une heure.
*.CT	Une mappe dans laquelle les entrées sont arrivées à expiration sur la base de la date/heure de création de l'entrée, plus la valeur TTL. La durée de vie par défaut est d'une heure.

Options de verrouillage

Tableau 2. Options de verrouillage de mappe dynamique

Option de verrouillage	Description
(vide)	Si vous indiquez pas d'option de verrouillage, aucun mécanisme de verrouillage n'est utilisé.
.P	Indique que la mappe possède le verrouillage pessimiste
.O	Indique que la mappe possède le verrouillage optimiste

2.5+

Options d'invalidation du cache local

Tableau 3. Options d'invalidation du cache local

Option de verrouillage	Description
(vide)	Si vous n'indiquez aucune valeur, l'invalidation du cache local est désactivée.
.NCI	Indique que la mappe utilise l'invalidation du cache local. Votre mappe doit être configurée avec le verrouillage optimiste ou sans aucun verrouillage.

Rubrique parent : Référence

Concepts associés:
Expulseurs

Tâches associées:
Configuration d'un expulseur TTL (Time To Live)
Configuration des mappes dynamiques
Java 2.5+ Configuration de l'invalidation du cache local
Développement d'applications de grille de données avec la passerelle REST
Création de grilles de données simples
.NET Création de mappes dynamiques avec les API .NET

CLI command reference

This page describes the command line interface (CLI) commands provided by WebSphere® DataPower® XC10 Appliance. The following sets of commands are supported:

- Common commands that work with various configuration objects. See Common commands.
- Special purpose commands. See Special purpose commands.

Common commands

Commands described in this section are used to work with various configuration objects. Common commands either work on a specified configuration object or are used within an edit session on a configuration object. An edit session is a mode of the CLI in which you can modify the values of a configuration object. An edit session starts with the "edit" or "create" commands, and ends with the "exit" or "cancel" commands. Some of these commands, such as "append" or "set", are available only within an edit session.

Currently supported configuration objects include:

- aggregate-interface
- ethernet-interface
- ldap-auth
- logging-settings
- netsec-settings
- network-settings
- vlan-interface

Commands that work within an edit session include the following commands:

- cancel
- create
- delete
- disable
- domain list
- domain show
- domain switch
- edit
- enable
- exit
- list
- reset
- show
- show properties
- show types
- status

Parent topic: CLI command reference

cancel

Purpose

Within an edit session, discard all changes that were made to a configuration object during the edit session or during a nested edit session.

Syntax

cancel

Parameters

None.

Usage Notes

In a nested edit session, cancel discards only the changes that were made in the edit submode.

Related Commands

See exit .

Example

Cancel changes that were made during an edit session on the network-settings configuration object.

```
Console> edit network-settings
Console network-settings> set tcp-retries 10
Console network-settings> cancel
Cancelled
Console>
```

Parent topic: Common commands

create

Purpose

Enter an edit session to create an instance of a configuration object.

Syntax

create *object-type object-instance*

Parameters

object-type
: Type of the object to be created

object-instance
: Name of the object instance to be created

Usage Notes

- Some objects cannot be created in the Command Line Interface. These objects include the following:
  - Objects for which there is only one instance, called singleton objects
  - Objects for which there is a predefined, finite set, such as the `ethernet-interface` objects.
- By default, the AdminState of a newly created object is enabled. .

Related Commands

See delete, edit, and list .

Example

Create a vlan-interface named v1000.

```
Console> create vlan-interface v1000
Console vlan-interface:v1000>
...
```

Parent topic: Common commands

delete

Purpose

Delete an instance of a configuration object.

Syntax

delete *object-type object-instance*

Parameters

object-type
: Type of the object to be deleted

object-instance
: Name of the object instance to be deleted

Usage Notes

Some objects cannot be deleted. These objects include the following:

- Objects for which there is only one instance, called singleton objects
- Objects for which there is a predefined, finite set, such as the `ethernet-interface` objects.

Related Commands

See create, edit, and list.

Example

Delete a vlan-interface named v1000.

```
Console> delete vlan-interface v1000
Console>
```

Parent topic: Common commands

disable

Purpose

- Disable a configuration object or an instance of a configuration object.
- This command changes the administrative state (AdminState) to "disabled". If no errors occur, the operational state (OpState) changes to "down".
- No other configuration values are changed as a result of this command.

Syntax

disable *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be disabled

object-instance
: Name of the object instance to be disabled, if the configuration object is not a singleton object

Usage Notes

Configuration changes can be made to an object that is administratively disabled, but the changes will not be applied until the object is enabled.

Related Commands

See enable.

Example

Disable a vlan-interface named v1000.

```
Console> disable vlan-interface v1000
Console>
```

Parent topic: Common commands

domain list

Purpose

Display the supported domains.

Syntax

domain list

Parameters

None.

Usage Notes

Currently only one domain, "default", is supported.

Related Commands

See domain show and domain switch.

Example

Show currently supported domains.

```
Console> domain list
default
Console>
```

Parent topic: Common commands

domain show

Purpose

Display the currently active domain.

Syntax

domain show

Parameters

None.

Usage Notes

Currently only one domain, "default", is supported.

Related Commands

See domain list and domain switch.

Example

Show currently active domain.

```
Console> domain show
In domain "default"
Console>
```

Parent topic: Common commands

domain switch

Purpose

Change the currently active domain.

Syntax

domain switch *domain*

Parameters

domain
: Required active domain

Usage Notes

Currently only one domain, "default", is supported.

Related Commands

See domain list and domain show.

Example

Change the currently active domain.

```
Console> domain switch default
Switched to "default" domain
Console>
```

Parent topic: Common commands

edit

Purpose

Enter a configuration mode to modify a configuration object or an instance of a configuration object.

Syntax

edit *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be edited

object-instance
: Name of the object instance to be edited, if the configuration object is not a singleton object

Usage Notes

The edit command puts you into an edit session for the configured object. Within this edit session, edit subcommands allow you to modify values for the properties of the configured object.

Related Commands

See create, delete, and list.

Example

Edit ethernet-interface eth3 to set the MTU to 4096.

```
Console> edit ethernet-interface eth3
Console ethernet-interface:eth3> set mtu 4096
Console ethernet-interface:eth3> exit
Console>
```

Parent topic: Common commands

enable

Purpose

- Enable a configuration object or an instance of a configuration object.
- This command changes the administrative state (AdminState) to "enabled". If no errors occur, the operational state (OpState) changes to "up".
- No other configuration values are changed as a result of this command.

Syntax

enable *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be enabled

object-instance
: Name of the object instance to be enabled, if the configuration object is not a singleton object

Related Commands

See disable .

Example

Enable a vlan-interface named v1000.

```
Console> enable vlan-interface v1000
Console>
```

Parent topic: Common commands

exit

Purpose

Leave edit mode or an edit submode.

Syntax

exit

Parameters

None.

Usage Notes

Configuration changes are saved when the edit session is ended, not on leaving an edit submode. This might cause validation errors to be displayed later than expected.

Related Commands

See cancel .

Example

Enter an edit session for an `ethernet-interface` object and enter the ip submode. Add an IP address and then exit the ip submode and exit the edit session.

```
Console> edit ethernet-interface eth2
Console ethernet-interface:eth2> ip
Entering "ip" mode
Console ethernet-interface:eth2 ip> append address 192.168.0.200/24
Console ethernet-interface:eth2 ip> exit
Console ethernet-interface:eth2> exit
Console>
```

Parent topic: Common commands

list

Purpose

Show the names of configured objects of a specified object type.

Syntax

list *object-type*

Parameters

object-type
: Type of the object to be listed

Usage Notes

Use show types to get the list of object types.

Related Commands

See create, delete, edit, and show types.

Example

List all the instances of ethernet-interfaces.

```
Console> list ethernet-interface
        eth0
        eth1
        eth2
        eth3
Console>
```

Parent topic: Common commands

reset

Purpose

Restore a configuration object or a part of a configuration object to its default settings.

Syntax

- Reset a singleton configuration object: reset *object-type*
- Reset an instance of a configuration object: reset *object-type object-instance*
- Reset a field of a configuration object in an edit session: reset *property*

Parameters

object-type
: Type of the object to be reset

object-instance
: Instance of the object to be reset, if the configuration object is not a singleton object

property
: Field within a configuration object to be reset

Example

- Reset the IP addresses for ethernet-interface eth3.

```
Console> edit ethernet-interface eth3
Console ethernet-interface:eth3> ip
Entering "ip" mode
Console ethernet-interface:eth3 ip> reset address
Console ethernet-interface:eth3 ip> exit
Console ethernet-interface:eth3> exit
Console>
```

- Reset the entire configuration object for ethernet-interface eth3 to default values.

```
Console> reset ethernet-interface eth3
Console>
```

Parent topic: Common commands

show

Note: The show command is used to display both configuration objects and other specific data. This section documents the use of the show command for configuration objects. See also the show commands within the Special purpose commands section.

Purpose

Show the configured values for a specified configuration object or an instance of a configuration object.

Syntax

show [*object type* [*object instance*]]

Parameters

object type
: Type of the object to be displayed

object instance
: Name of the object instance to be displayed, if the configuration object is not a singleton object

Usage Notes

- Within an edit session, the show command displays the currently configured values for the object instance that is being edited.
- The operational state of the object is also displayed. Only properties that are required and properties that have had values configured differently from the assigned default values are displayed. Optional properties that have not been explicitly configured will not be displayed.
- The values displayed here are the desired configuration for the object. In contrast, the status command will display the actual runtime values for the object.

Example

Show the configured values for the network-settings (singleton) object.

```
Console> show network-settings
network-settings : [Up]

name "network-settings"
AdminState "Enabled"
icmp-options " "
explicit-congestion-notification "false"
destination-based-routing "false"
interface-isolation "relaxed"
tcp-retries "5"
arp-retries "8"
arp-retry-interval "500"
tso-offload "true"
reverse-path-filtering "false"
tcp-window-scaling "true"
```

Parent topic: Common commands

show properties

Purpose

Within an edit session, display the configurable properties of the object.

Syntax

show properties

Parameters

None.

Usage Notes

To see the properties of a nested level, enter the edit submode and repeat the command.

Example

Display the configurable properties of an ethernet-interface.

```
Console> edit ethernet-interface eth3
Console ethernet-interface:eth3> show properties
Basic properties:
        AdminState
        mac-address
        mode
        mtu
        use-arp
        userdata

Sub-mode (complex) properties:
        ip

Console ethernet-interface:eth3> ip
Entering "ip" mode
Console ethernet-interface:eth3 ip> show properties
Basic properties:
        dad-retransmit-timer
        dad-transmits
        use-dhcp
        use-slaac

Array (list) properties:
        address
        static-route

Sub-mode (complex) properties:
        ipv4-default-gateway
        ipv6-default-gateway

Console ethernet-interface:eth3 ip>
```

Parent topic: Common commands

show types

Purpose

Display the types of configuration objects that are supported on the appliance. This does not reflect any instances that are configured, but just types that are supported.

Syntax

show types

Parameters

None.

Usage Notes

None.

Example

Display the configuration objects on the appliance.

```
Console> show types
The following types are defined:
        aggregate-interface
        ethernet-interface
        logging-settings
        network-settings
        vlan-interface
```

Parent topic: Common commands

status

Note: The status command is used to display both status of configured objects and specific functional status information. This section documents the use of the status command for configuration objects. See also the status commands within the Special purpose commands section.

Purpose

Show the status of a specified configuration object or an instance of a configuration object.

Syntax

status *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be displayed

object-instance
: Name of the object instance to be displayed, if the configuration object is not a singleton object

Usage Notes

- If the *object-instance* is omitted for a configuration object that is not a singleton object, the status of all instances of the specified *object-type* is displayed.
- The status command is supported only for selected configuration objects.
- The values displayed here are the runtime values for the object. In contrast, the show command will display the configured properties of the object.

Example

Show the status of ethernet-interface eth0.

```
Console> status ethernet-interface eth0
eth0     OpState:[Up]
        generic MTU:1500 carrier:true flags:UP BROADCAST RUNNING MULTICAST
            index:5
        inet addr:9.42.77.48 flags:PERMANENT mask:255.255.255.0
            scope:GLOBAL
        inet6 addr: 2002:92a:8f7a:901:9:42:77:48 flags:PERMANENT
            mask: ffff:ffff:ffff:ffff:: scope:GLOBAL
        inet6 addr: fe80::20c:29ff:fedd:7baf flags:PERMANENT
            mask: ffff:ffff:ffff:ffff:: scope:LINK
        ethernet Link:on MAC: 00:0c:29:dd:7b:af autoneg:on duplex:Full
            port:TP speed:1000Mbps
        statistics collisions:0 multicast:0 rx_bytes:69088545
            rx_compressed:0 rx_crc_errors:0 rx_dropped:0 rx_errors:0
            rx_fifo_errors:0 rx_frame_errors:0 rx_length_errors:0
            rx_missed_errors:0 rx_over_errors:0 rx_packets:842919
            tx_aborted_errors:0 tx_bytes:3512656 tx_carrier_errors:0
            tx_compressed:0 tx_dropped:0 tx_errors:0 tx_fifo_errors:0
            tx_heartbeat_errors:0 tx_packets:18144 tx_window_errors:0
```

Parent topic: Common commands

Special purpose commands

Commands described in this section take a specific action on the appliance or display specific information.

- packet-capture start
- packet-capture stop
- platform collect-pd
- platform log-level get
- platform log-level set
- platform service disable ssh
- platform service disable telnet
- platform service enable ssh
- platform service enable telnet
- platform must-gather
- raid delete
- raid re-use
- raid foreign-config import
- raid foreign-config clear
-  request force-suspend
-  request resume
-  request suspend
- set-dns-search
- set-dns-servers
- set-ntp-servers
- show commandref
- show components
- show locales
- show log
- show timezones
- show version
- sshkey fingerprint
- sshkey reset
- sshkey verify
- start-progress
- status battery
- status cpu-usage
- status cpu-utilization
- status dynamic-tunnels
- status fan
- status flash
- status intrusion
- status memory
- status nvdimm
- status power-supply
- status raid all
- status raid battery
- status raid physical
- status voltage
- status volume
- status temperature
- status uptime
- timezone get
- timezone set
- unalias
- usage
- user add
- user delete
- user known-hosts list
- user known-hosts delete
- user list
- user password
- user sshkey add
- user sshkey delete

Parent topic: CLI command reference

add-jvm-args

Purpose

Adds a predetermined set of JVM configuration parameters to any of the processes on the appliance during startup.

Syntax

add-jvm-args process name listOfOneOrMoreJVMArgs

Parameters

There are two required parameters for this command:

process name
: Name of the process that needs to be configured. Valid processName arguments include:

  - all - All processes, including xsa.app, catalog server and all container servers.
  - console - Includes the xsa.app process only.
  - grids - All container server processes only.
  - gridNN - Where NN equals a number (i.e 01) that specifies a single container server.

listOfOneOrMoreJVMArgs
: List valid JVM arguments for the process. Valid JVM arguments you can specify include:

  - -Dcom.ibm.CORBA.Debug
  - -Dcom.ibm.CORBA.CommTrace
  - -Djavax.net.debug
  - -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true
  - -verbose:gc
  - -Xverbosegclog
  - -Xhealthcenter
  - -Xcheck:jni

  Note: If there is more than one parameter you want to include in the list, separate with `!:!`, for example: `add-jvm-args grid01 -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true!:!`

  `-Dcom.ibm.CORBA.CommTrace add-jvm-args all -Dcom.ibm.CORBA.CommTrace`

Related Commands

See device restart.

Example

```
Console> add-jvm-args grid01 -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true
Updating JVM Args to include new args...
Adding the following JVM Arg: -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true
[Wed Apr 10 2013 17:00:22]  Run device restart to enable the new JVM args settings.
```

Parent topic: Special purpose commands

alias

Purpose

- Define an alternative name or a shortcut for a command line interface (CLI) command.
- Display configured aliases.

Syntax

- Display the list of defined aliases: alias
- Display the definition for an existing alias: alias *name*
- Define an alias for a specified CLI command: alias *name value*

Parameters

name
: Alternative name for a command line interface (CLI) command.

value
: Substitute value for the specified name. If the substitute value contains spaces, enclose the value in double quotation marks.

Related Commands

See unalias.

Example

Define a shortcut for testing connectivity to a particular address. Then display the configured aliases.

```
Console> alias test-connectivity "net-test ping 9.42.106.2"
Console> alias
alias test-connectivity "net-test ping 9.42.106.2"
Console>
```

Parent topic: Special purpose commands

clear-all

Purpose

Resets the configuration data for the appliance, including removing all the data grids and new users. All of the internal processes that are running on the appliance are restarted. This command runs without restarting the appliance hardware.

Syntax

clear-all

Parameters

None.

Related Commands

None.

Example

```
Console> clear-all
Force Stopped all XC-10 processes
Deleting configuration data and logs
Deleting grid data
```

Parent topic: Special purpose commands

clear-jvm-args

Purpose

Clears JVM configuration parameters to any of the processes that had been applied to the appliance during startup.

Syntax

clear-jvm-argsprocessName

Parameters

process name
: Name of the process that needs to be configured. Valid processName arguments include:

  - all - All processes, including xsa.app, catalog server and all container servers.
  - console - Includes the xsa.app process only.
  - grids - All container server processes only.
  - gridNN - Where NN equals a number (i.e 01) that specifies a single container server.

Related Commands

See add-jvm-args.

Example

```
Console> clear-jvm args all
```

Parent topic: Special purpose commands

clear-logs

Purpose

Deletes all of the WebSphere® DataPower® XC10 Appliance log files.

Syntax

clear-logs

Parameters

None.

Related Commands

None.

Example

```
Console> clear-logs
Cleared All Logs
```

Parent topic: Special purpose commands

clear-tls-config

Purpose

Resets the Transport Layer Security (TLS) configuration. Run this command if clear-all is not an option because you do not want to lose all configuration data, but TLS configuration becomes corrupted or you want to restore the default TLS values. In most situations, use the user interface for TLS changes or use the clear-all command to revert to the original configuration of the appliance. Run the clear-tls-config command on each appliance in the collective. After running the command on each appliance, restart the processes in each appliance in the collective. If the collective is successfully communicating, use the device restart command. The collective is communicating properly when all appliances in the collective are accessible through the user interface and can be seen as started in the Collective panel. However, if the TLS configuration is preventing the collective from communicating and the device restart command does not bring the appliance back up, you can use the force-recycle command to forcibly stop and start all the processes in the appliance without saving any data.

Syntax

clear-tls-config

Parameters

None.

Related Commands

clear-all, device restart, and force recycle.

Example

```
Console> clear-tls-config
Cleared Transport Layer Security Configuration
Run clear-tls-config on the other appliances in the collective and restart each appliance for chan
ges to take effect
```

Parent topic: Special purpose commands

collect-logs

This command has been deprecated. Use platform must-gather instead.
Parent topic: Special purpose commands

component firmware update

Purpose

Update component firmware version to configured version if current firmware version is not matched.

Syntax

component firmware update component type

Parameters

component type

- component type name. see help for this command to get a full list of supported component types.
- when using show components and if a component firmware version is not matched with the configured version, you can use this command to update its firmware version.

Related Commands

See show components.

Example

Update component firmware for BMC.

```
Console> component firmware update bmc
CWZBR03003I: Are you sure you want to update component firmware for bmc
After update, device will reboot
CWZBR03012I: yes/no:yes
CWZBR03013I: Component firmware update is starting.....
CWZBR03006I: Component firmware update succeeded and reboot device
```

Parent topic: Special purpose commands

config

Purpose

Import or export appliance configurations.

Syntax

```
config <export|import|usage|listExports> [-file] <filename> [-keyStore] <keystore>
[-trustStore] <truststore> [-silent] [-noRestart]
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

-truststore
: The -trustStore flag is followed by the truststore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-keystore
: The -keyStore flag is followed by the keystore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-noRestart
: Certain appliance settings require you to restart your appliances. If you set this flag, then none of your appliances are restarted. However, if you set the -noRestart parameter, then you must manually restart your appliances to preserve settings that require it.

-silent
: When specified, the config command status messages are not displayed, which is the default behavior.

Usage notes

None

Related commands

This command has the following subcommands: See usage, export, import, listExports, and deleteExport.

Example

Import or export appliance configurations.

```
config export -file foo.json -silent
config import -file foo.json -keyStore keyStore.jks -trustStore trustStore.jks
config import -file foo.json -noRestart
config usage
config listExports
config deleteExport export_1942-09-06_13-50-14.036.json
```

Parent topic: Special purpose commands

datetime get

Purpose

Display the time that is configured on the appliance, both in Greenwich mean time (GMT) and local time.

Syntax

datetime get

Parameters

None.

Related Commands

See datetime set, timezone get, and timezone set.

Example

Display the configured time on the appliance.

```
Console> datetime get
GMT: 2012-03-08T20:20:21Z
Local: Mar 8, 2012 3:20:21 PM
Console>
```

Parent topic: Special purpose commands

datetime set

Purpose

Configure the current time on the appliance.

Syntax

datetime set *time*

Parameters

time

- Current time in local time (as determined by the timezone setting).
- Format is `YYYY-MM-DD hh:mm:ss` or `YYYY-MM-DD hh:mm:ssZ` with the following specifications:

  `YYYY`
  : Represents the year

  `MM`
  : Represents the month

  `DD`
  : Represents the day of the month

  `hh`
  : Represents hour

  `mm`
  : Represents minutes

  `ss`
  : Represents seconds

  `Z`
  : Appends to the time to specify it in Greenwich mean time (GMT).

Related Commands

See datetime get, timezone get, and timezone set.

Example

Display the configured timezone, set the current time on the appliance and then display configured time.

```
Console> timezone get
Timezone is EST
Console> datetime set 2012-03-22 10:01:00
Console> datetime get
GMT: 2012-03-22T15:01:03Z
Local: Mar 22, 2012 10:01:03 PM
Console>
```

Parent topic: Special purpose commands

deleteExport

Purpose

Delete a specified export file from the list of exports that are stored locally.

Syntax

```
config deleteExport
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

Usage notes

You must pass the deleteExport command and specify only one file with the -file flag.

Example

Delete exported appliance configurations.

```
config deleteExport export_1942-09-06_13-50-14.036.json
```

Parent topic: Special purpose commands

device RESET

Purpose

Erase all data on the appliance and put it back to a factory-new state. All configuration is deleted, including the ssh server keys, license acceptance, and logs. All user IDs and passwords are also deleted and replaced with the default user ID and password that were set at the factory. The appliance will automatically reboot after you run this command.

The "RESET" part of the command is required to be all uppercase to lower the probability of it being typed accidentally. This command differs from firmware pristine-install because this command does not change the firmware that is running on this device. Therefore, you do not need a firmware image when you run this command. This command changes only the configuration.

Syntax

device RESET [noprompt]

Options

noprompt
: Do not prompt for confirmation. The command is started immediately. If this option is not present, the command is not started unless the user answers affirmatively to a confirmation prompt that follows.

Parameters

None.

Usage Notes

If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is ended when the configuration is cleared. If you want to see all the messages for this command, you need to be connected to the serial console.

Related Commands

See firmware pristine-install.

Example

```
Console> device RESET

This command will result in the network configuration being reset; this may end your
current connection to the appliance.

Undo all existing configuration and reset the appliance to factory-new condition? y/n: y
You have chosen to reset the device, proceeding with reset...
Reset complete; ready to reboot
Console>
```

Parent topic: Special purpose commands

device battery-replaced

Purpose

Log the date on which the internal battery was replaced. The battery provides backup power for the hardware configuration and RAID disk writeable cache when the appliance is powered off and unplugged from an AC power source.

Batteries are valid for a specified duration, and this command marks the beginning of that duration. This command was run at the factory, and needs to be run by a user only when a battery is replaced in the field. The current installation date and expiration date of the existing battery can be shown with the status battery command.

Syntax

device battery-replaced

Options

None.

Parameters

None.

Related Commands

See status battery.

Example

```
Console> device battery-replaced
Ok
Console>
```

Parent topic: Special purpose commands

device intrusion allow

Purpose

- Allow the case to be opened on those devices that allow it.
- See device intrusion clear.

Syntax

device intrusion allow

Parameters

None.

Example

```
Console> device intrusion allow
Ok
Console>
```

Parent topic: Special purpose commands

device intrusion clear

Purpose

- Clear device of intruded state recorded when the appliance case is physically opened so that the case is no longer considered intruded.
- Some customer-replaceable or field-replaceable parts on the appliance might require the case to be opened. Run this command after the parts are replaced and the case is closed.

Syntax

device intrusion clear

Parameters

None.

Usage Notes

- This command is valid only on a 7198 or 7199 machine types.
- Opening the case on a 9235 machine type will render the appliance permanently inoperable, requiring that the appliance be returned to the factory, so do not open the case on a 9235 machine type. These physical appliances have sensors that detect when the case has been opened. On other machine types of physical appliances, this command will not have any effect. This command is not applicable on virtual appliances.
- See also status intrusion.

Example

```
Console> device intrusion clear
Ok
Console>
```

Parent topic: Special purpose commands

device intrusion disallow

Purpose

Reenable intrusion detection after a previous invocation of device intrusion allow.

Syntax

device intrusion disallow

Parameters

None.

Usage Notes

- This command has effect only on machine types 7198 and 7199. See device intrusion clear for more information.
- This command does not reset the intrusion sensor. It is generally a good idea to invoke device intrusion clear before rebooting the appliance.

Example

```
Console> device intrusion disallow
Ok
Console>
```

Parent topic: Special purpose commands

device nvdimm clear

Purpose

Clear the nvDIMM volume labels.

Syntax

device nvdimm clear

Parameters

None.

Usage Notes

This command is only allowed on 9006 machine types.

Related Commands

None.

Example

```
Console> device nvdimm clear
CWZBR02923I:  Volume labels cleared
Console>
```

Parent topic: Special purpose commands

device clear-intrusion

This command is deprecated. Use device intrusion clear instead.
Parent topic: Special purpose commands

device restart

Purpose

Shut down the appliance and reboot it. On reboot, all configured applications start. This is the preferred method to reboot the appliance instead of power cycling it.

Syntax

device restart

Options

None.

Parameters

None.

Usage Notes

If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is ended during the shutdown. If you want to see all the messages for the full restart cycle, you need to be connected to the serial console.

Related Commands

See device shutdown.

Example

```
Console> device restart
Ok
Console>
Preparing system for shutdown: s1  hb  hn  2  0  s2  rs  nt  hc  ns  sy  ul r6
machine restart
Starting system... dn un nvm 7198 r1 r2 lo wf lu pt tc sr in cd ln mu ku up us
t1 t2 md mm mcg mt ml na fi
login:
```

Parent topic: Special purpose commands

device shutdown

Purpose

Shut down the appliance and leave it in a powered-off state. It does not automatically reboot. If you want to reboot the appliance after running this command, push the power button if it is a single push button, or toggle the power button if it is a 2-position rocker switch. This is the preferred method to shut down the appliance instead of using only the power button.

Syntax

device shutdown

Options

None.

Parameters

None.

Usage Notes

If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is ended during the shutdown. If you want to see all the messages for the full shutdown cycle, you need to be connected to the serial console.

Related Commands

See device restart.

Example

```
Console> device shutdown
Ok
Console>
Preparing system for shutdown:  s1  hb  hn  2  0  s2  rs  nt  hc  ns  sy  ul  r0
machine shutdown
```

Parent topic: Special purpose commands

echo

Purpose

Write to the output stream of the command line interface (CLI).

Syntax

echo *text*

Parameters

text
: Text to be written to the stream of the command line interface (CLI). It is displayed only in the command line interface (CLI) instance from which this command was issued.

Usage Notes

The echo command is useful in command line interface (CLI) scripts.

Example

Use the echo command to prepend output of the datetime get display.

```
Console> echo Current time is: ; datetime get
Current time is:
GMT: 2012-03-22T15:32:06Z
Local: Mar 22, 2012 3:32:06 PM
Console>
```

Parent topic: Special purpose commands

export

Purpose

Export the appliance configuration either locally or to a temporary location for uploading to a remote file server.

Syntax

```
config export
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

Usage notes

You must pass the -file flag on the `export` command.

Example

Export the appliance configuration.

```
config export -file foo.json -silent
```

Parent topic: Special purpose commands

file delete

Purpose

Delete a file from the appliance.

Syntax

file delete *filename*

Parameters

filename
: File to be deleted

Related Commands

See file get, file list, and file put.

Example

Delete a file that is previously created by the platform collect-pd command.

```
Console> file list
collect-pd.txt 103576 bytes created Mar 22, 2012 3:38:32 PM
Console> file delete collect-pd.txt
Console>
```

Parent topic: Special purpose commands

file get

Purpose

Copy a file to the appliance from a remote location.

Syntax

file get *url localfile*

Parameters

url
: Uniform resource locator that is specified in one of the two following formats. In the following formats, white space is added only for readability. Do not included any white space in the URL on the command.

- For protocols HTTP and FTP:

  protocol`: // [`*username* `[:`*password*`] @]`*hostname* `[:`*port*`]/[%2F` *abspath*`/][`*relpath* `/]` *filename*

  where

  protocol
  : FTP or HTTP.

  username
  : User name at the remote host. This is an optional component of URL.

  password
  : Password for *username* at the remote host. This is an optional component of URL.

  hostname
  : Remote host name or IP address.

  port
  : Decimal port number of the remote server. This is an optional component of URL.

  abspath
  : Absolute path for the remote file, prefixed with the 3 characters '%2F' or '%2f'. It is supported for protocol FTP and SCP and is an optional component of URL.

  relpath
  : Relative path for the remote file. This path is relative to the default directory on the remote system. If *username* is specified, the default directory is typically the home directory of that user. This path is an optional component of URL.

  filename
  : Remote file name.

- For protocols SCP:

  protocol`: //` *username*`@`*hostname*`: [%2F` *abspath*`/][`*relpath* `/]` *filename*

  where

  protocol
  : SCP.

  username
  : User name at the remote host. This is an optional component of URL.

  hostname
  : Remote host name or IP address. Host names that resolve to IPv6 addresses are supported; however, literal IPv6 addresses, which contain colons, are not supported because the *hostname* is separated from the file path with a colon.

abspath
: Absolute path for the remote file, prefixed with the 3 characters '%2F' or '%2f'. It is supported for protocol FTP and SCP and is an optional component of URL.

relpath
: Relative path for the remote file. This path is relative to the default directory on the remote system. If *username* is specified, the default directory is typically the home directory of that user. This path is an optional component of URL.

filename
: Remote file name.

localfile
: Name to be assigned to the local copy of the file. No absolute path elements (leading forward slashes) or subdirectories (forward slashes) are allowed. It must be a relative, flat name.

Related Commands

See file delete, file list, and file put.

Example

Retrieve a firmware file from a remote location and give it the name of `newfirmware` on the appliance.

```
Console> file get scp://user1@server1.raleigh.ibm.com:~/dev_bedrock.scrypt2 newfirmware.scrypt2
user1@server1.raleigh.ibm.com's password:
dev_bedrock.scrypt2                           100%  136MB  34.0MB/s   00:04
Wrote 142553856 bytes to local storage
Console>
```

Parent topic: Special purpose commands

file list

Purpose

List the files on the appliance.

Syntax

file list

Parameters

None.

Related Commands

See file delete, file get, and file put.

Example

Show the files that are currently on the appliance.

```
Console> file list
collect-pd.txt 103794 bytes created Mar 22, 2012 5:30:56 PM
newfirmware 142553856 bytes created Mar 22, 2012 4:41:17 PM
Console>
```

Parent topic: Special purpose commands

file put

Purpose

Copy a file to a remote location from the appliance.

Syntax

file put *localfile url*

Parameters

localfile
: Name of the local file to be transferred. This file must exist.

url
: Uniform resource locator, as described in file get; however, the HTTP protocol is not supported for the file put command.

Related Commands

See file delete, file get, and file list.

Example

Copy local file `collect-pd.txt` to a remote system as `diag.txt`.

```
Console> file put collect-pd.txt scp://user1@system1.rtp.raleigh.ibm.com:~/diag.txt
user1@system1.rtp.raleigh.ibm.com's password:
collect-pd.txt                                100%  101KB 101.4KB/s   00:00
Console>
```

Parent topic: Special purpose commands

firmware pristine-install

Purpose

Install firmware and restore the appliance to a near-factory-fresh state.

Syntax

firmware pristine-install *image*

Parameters

image
: Firmware image to be installed, in the form of an scrypt2 file

Usage Notes

- The pristine-install command installs the new firmware image without copying over configuration or data files copied over on a typical firmware upgrade.
- The pristine-install command can be used to install an older version of firmware on the appliance.
- The pristine-install command can also be used to attempt recovery in error situations where less disruptive methods do not succeed.
- The appliance is rebooted as part of this operation, so network connectivity is disrupted. Therefore, it is recommended that this command be typically run from the serial console.

Related Commands

See firmware upgrade, firmware rollback, and device RESET.

Example

Revert the firmware to an older firmware level.

```
Console> firmware pristine-install oldfirmware.scrypt2
Upgrading firmware...
Verifying image signature
Executing dynamic loader
Executing dynamic loader
Validating image
Extracting firmware from image
Extracting firmware manifest
Executing pre-installation
Deleting previous installation
Linking common files.
Extracting files
Verifying installation
Copying configuration from existing installation
Switching to new installation
Upgrade or rollback succeeded. Rebooting
```

Parent topic: Special purpose commands

firmware rollback

Purpose

Revert the firmware level to the previous level of firmware that is installed on the appliance.

Syntax

firmware rollback

Parameters

None.

Usage Notes

- The appliance holds at most two levels of firmware, the active one and the alternate. Successive invocations of the firmware rollback command result in switching back and forth between the two images.
- This command rolls back the configuration. After the rollback, the configuration is in the state it was in before the last upgrade.
- This command rolls back the component firmware. Because each component can have only one copy of component firmware, the component firmware from the version that is rolled back is reinstalled to the component.
- The appliance is rebooted as part of this operation, so network connectivity is disrupted. Therefore, it is recommended that this command be typically run from the serial console.

Related Commands

See firmware upgrade, firmware pristine-install, and device RESET.

Example

Revert the appliance to the previous version of firmware that is installed.

```
Console> firmware rollback
Rolling back firmware...
Upgrade or rollback succeeded. Rebooting...
```

Parent topic: Special purpose commands

firmware upgrade

Purpose

Install new level of firmware on the appliance.

Syntax

firmware upgrade *image*

Parameters

image
: Firmware image to be installed, in the form of an scrypt2 file

Usage Notes

The appliance is rebooted as part of this operation, so network connectivity is disrupted. Therefore, it is recommended that this command be typically run from the serial console.

Related Commands

See firmware pristine-install, firmware rollback, and device RESET.

Example

Install a new level of firmware on the appliance.

```
Console> firmware upgrade myfirmware.scrypt2
Upgrading firmware...
Verifying image signature
Executing dynamic loader
Validating image
Extracting firmware from image
Extracting firmware manifest
Executing pre-installation
Deleting previous installation
Linking common files
Extracting files
Verifying installation
Copying configuration from existing installation
Switching to new installation
Upgrade or rollback succeeded. Rebooting
```

Parent topic: Special purpose commands

force recycle

Purpose

Restarts the WebSphere DataPower XC10 Appliance processes without saving any data. Because data loss can occur, run this command only if you are not worried about data loss or you have tried the device restart command and the appliance did not become available.

Syntax

force recycle

Parameters

None.

Related Commands

device restart

Example

```
Console> force-recycle
Force Stopped all XC-10 processes
Forced recycle of appliance
```

Parent topic: Special purpose commands

get-dns-search

Purpose

Display configured domain name server search domains.

Syntax

get-dns-search

Parameters

None.

Related Commands

See set-dns-search, get-dns-servers, and set-dns-servers.

Example

Configure and then display domain name server search domains.

```
Console> set-dns-search short.example.com example.com
Ok
Console> get-dns-search
DNS search domains:
        short.example.com
        example.com
Console>
```

Parent topic: Special purpose commands

get-dns-servers

Purpose

Display configured domain name server IP addresses.

Syntax

get-dns-servers

Parameters

None.

Related Commands

See set-dns-servers, get-dns-search, and set-dns-search.

Example

Configure and then display domain name servers.

```
Console> set-dns-servers 9.42.106.2  2002:92a:8f7a:106:9:42:106:42
Ok
Console> get-dns-servers
Domain (DNS) servers:
        9.42.106.2
        2002:92a:8f7a:106:9:42:106:42
Console>
```

Parent topic: Special purpose commands

get-ntp-servers

Purpose

Display configured network time protocol servers.

Syntax

get-ntp-servers

Parameters

None.

Related Commands

See set-ntp-servers.

Example

Configure and then display NTP servers.

```
Console> set-ntp-servers example.server.com 127.0.0.1
Ok
Console> get-ntp-servers
NTP servers:    example.server.com
                127.0.0.1
Console>
```

Parent topic: Special purpose commands

help

Purpose

Display command help information.

Syntax

- Display list of supported commands: help command
- Display help for a specific command: help *command*

Parameters

command
: Command line interface (CLI) command

Example

Display subcommands for the platform command.

```
Console> help platform
The following platform commands are available:
        platform collect-pd [PDFfilename | console]
        platform log-level ...
        platform must-gather filename [PDFfilename]
```

Parent topic: Special purpose commands

import

Purpose

Import the appliance configuration.

Syntax

```
config import
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

-truststore
: The -trustStore flag is followed by the truststore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-keystore
: The -keyStore flag is followed by the keystore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-noRestart
: Certain appliance settings require you to restart your appliances. If you set this flag, then none of your appliances are restarted. However, if you set the -noRestart parameter, then you must manually restart your appliances to preserve settings that require it.

-silent
: When specified, the config command status messages are not displayed, which is the default behavior.

Usage notes

You must pass the import command with the -file flag.

Example

Import the appliance configuration.

```
config import -file foo.json -silent
```

Parent topic: Special purpose commands

license accept

Purpose

Display licenses and require the command line interface (CLI) user to accept or reject licenses.

Syntax

license accept

Parameters

None.

Example

Run the license acceptance process.

```
Console> license accept
Bedrock Project, WebSphere Technical Institute
Copyright 2009-2010, IBM Corporation

This is your sample license.

Accept; Reject: accept
Console>
```

Parent topic: Special purpose commands

license get

Purpose

Transfer a new or changed license to the appliance.

Syntax

license get *url*

Parameters

url
: Remote location of license to be retrieved

Usage Notes

URL must be specified with one of the following protocols: HTTP, FTP, or SCP. See URL syntax for a description of the URL syntax.

Example

Retrieve new license `lic-test` to license repository.

```
Console> license get scp://user1@server1.raleigh.ibm.com:~/temp/lic
user1@server1.raleigh.ibm.com's password:
lic                                           100%   14     0.0KB/s   00:00
Wrote 14 bytes to local storage
Console>
```

Parent topic: Special purpose commands

license reset

Purpose

Reset the license acceptance status.

Syntax

license reset

Parameters

None.

Usage Notes

The next invocation of the command line interface (CLI) requires that the user accept licenses.

Example

Reset the license acceptance status.

```
Console> license reset
Ok
```

Parent topic: Special purpose commands

listExports

Purpose

Display a list of exported configurations that are stored locally.

Syntax

```
config listExports
```

Parameters

None

Usage notes

None

Example

List exported appliance configurations.

```
config listExports
```

Parent topic: Special purpose commands

locale get

Purpose

Retrieve the locale setting used to control display of messages in the command line interface (CLI).

Syntax

locale get

Parameters

None.

Example

Retrieve the locale setting.

```
Console> locale get
Locale is en_US
Console>
```

Parent topic: Special purpose commands

locale set

Purpose

Set the locale to be used in display of messages in the command line interface (CLI).

Syntax

locale set *language* [*country*]

Parameters

language
: Abbreviation for language

country
: Abbreviation for country

Usage Notes

The locale setting will take effect immediately for the CLI in which the command was issued. For other CLI instances that might already be active, the user must exit and reenter the CLI for the new locale setting to take effect.

Example

Set the locale to English by using the conventions of the United States.

```
Console> locale set en US
Console>
```

Parent topic: Special purpose commands

locate-led

Purpose

Turn on or off the locate LED. The locate LED is a light on the front of the physical appliance. It is labeled with an icon in the shape of a lighthouse, and the LED color is blue. The intent of the locate LED is to help customers visually locate a particular appliance in a datacenter. The locate LED is off by default, and must be explicitly turned on by using this command. The state of the LED is not persisted, so after an appliance is rebooted, its locate LED will be off. The locate LED has no function other than as a manual visual locator. This applies only to physical appliances. A virtual appliance has no locate LED.

Syntax

locate-led { on | off}

Options

None.

Parameters

on
: Turn on the locate LED

off
: Turn off the locate LED

Usage Notes

It is not possible to query the state of the locate LED. Setting the state of the locate LED to a state it is in is not considered an error.

Example

```
Console> locate-led on
Ok
Console>
```

Parent topic: Special purpose commands

log message

Purpose

- Write a text string into the diagnostic log file.
- For example, use this command to indicate the start or end of a particular activity. A manual review or automated post-processing of the log file can find the text and use it as a starting point for further analysis.
- This message is logged at the INFO (informational) level.

Syntax

log message *text*

Parameters

text
: Message text to write to the log

Example

```
Console> log message About to run the acceptance tests
Console>
```

Parent topic: Special purpose commands

netif

The netif commands are deprecated and are no longer present by default.

Use the commands documented in Common commands on the `ethernet-interface` configuration object instead.

Parent topic: Special purpose commands

net-test available

Purpose

Test whether a network interface has an active carrier.

Syntax

net-test available

Parameters

None.

Related Commands

For other diagnostic commands, see net-test dns, net-test ping, and net-test tcp.

Example

Test availability of an active network interface.

```
Console> net-test available
Network available
Console>
```

Parent topic: Special purpose commands

net-test dns

Purpose

Perform domain name server lookup on specified host name.

Syntax

net-test dns *hostname*

Parameters

hostname
: Name of host to use in domain name server search

Related Commands

See net-test available, net-test ping, and net-test tcp.

Example

Perform DNS lookup on system1.rtp.ibm.com.

```
Console> net-test dns system1.rtp.raleigh.ibm.com
9.42.77.42
Console>
```

Parent topic: Special purpose commands

net-test ping

Purpose

Test connectivity to a specified host name or IP address.

Syntax

net-test ping *host*

Parameters

host
: Host name or IP address of the remote system

Related Commands

For other diagnostic commands, see net-test available, net-test dns, and net-test tcp.

Example

Test connectivity to a specified host name. The remote system is reachable.

```
Console> net-test ping system1.rtp.raleigh.ibm.com
Ok
Console>
```

Test connectivity to a specified address. The remote system is not reachable.

```
Console> net-test ping 4.4.4.4
ping failed
Console>
```

Parent topic: Special purpose commands

net-test tcp

Purpose

Test ability to open a TCP connection to port at a specified host name or IP address.

If successfully opened, no data is sent across the TCP connection, it is closed.

Syntax

net-test tcp *host port*

Parameters

host
: Host name or IP address of the remote system

port
: TCP port at the remote system

Related Commands

For other diagnostic commands, see net-test available, net-test dns, and net-test ping.

Example

Test ability to reach port 80 at host system1.rtp.raleigh.ibm.com.

```
Console> net-test tcp system1.rtp.raleigh.ibm.com 80
Ok
```

Parent topic: Special purpose commands

net-test traceroute

Purpose

Report path to a specified hostname or IP address.

If successfully opened, no data is sent across the TCP connection, it is closed.

Syntax

net-test traceroute *host*

Parameters

host
: Host name or IP address of the remote system

Related Commands

For other diagnostic commands, see net-test available, net-test dns, net-test ping, and net-test tcp.

Example

Determine route to specified host.

```
Console> net-test traceroute localhost
localhost (127.0.0.1)  0.097 ms
Ok
Console>
```

```
Console> net-test traceroute 127.0.0.1
localhost (127.0.0.1)  0.099 ms
Ok
Console>
```

Test connectivity to specified address. Remote system is not reachable.

```
Console> net-test traceroute host.that.does.not.exist
bad address 'host.that.does.not.exist'
traceroute failed
Console>
Console> net-test traceroute 9.65.47.137
sendto: Network is unreachable
traceroute failed
```

Parent topic: Special purpose commands

nodename get

Purpose

Retrieve the configured name of an appliance.

Syntax

nodename get

Parameters

None.

Related Commands

See nodename set.

Example

Set and then retrieve the appliance name.

```
Console> nodename set appliance1
Console> nodename get
nodename is appliance1
Console>
```

Parent topic: Special purpose commands

nodename set

Purpose

Set the configured name of an appliance.

Syntax

nodename set *name*

Parameters

name
: Name of appliance

Related Commands

See nodename get.

Example

Set and then retrieve the appliance name.

```
Console> nodename set appliance1
Console> nodename get
nodename is appliance1
Console>
```

Parent topic: Special purpose commands

packet-capture clear

Purpose

Remove packet trace capture files for specified network interface.

Syntax

packet-capture clear *interface*

Parameters

interface
: Network interface for which packet trace files need to be cleared

Usage Notes

- After a packet trace is run on an interface, the files must be cleared before another packet capture can be started for the same interface.

Related Commands

See packet-capture list, packet-capture start, and packet-capture stop.

Example

Display packet capture activity and then clear the files collected for interface `eth0`.

```
Console> packet-capture list
interface       status          file(s)
eth1    Finished                eth1pc1
                                   eth1pc2
------------------------------
eth0    Finished                eth0pc
Ok
Console> packet-capture clear eth0
Packet capture dump files cleared for eth0
Ok
Console>
```

Parent topic: Special purpose commands

packet-capture list

Purpose

Show packet-capture activity on the appliance.

Syntax

packet-capture list

Parameters

None.

Related Commands

See packet-capture clear, packet-capture start, and packet-capture stop.

Example

Show packet-capture activity on the appliance. In this example, packet-capture has finished for interface **eth0**. For interface **mgt0** trace is still being actively written to three ring buffer files.

```
Console> packet-capture list
interface       status          file(s)
eth0    Finished                eth0pc
--------------------------------
mgt0    Capturing               mgt0pc1
                                mgt0pc2
                                mgt0pc3
--------------------------------
Ok
Console>
```

Parent topic: Special purpose commands

packet-capture start

Purpose

Activate packet trace capture.

Syntax

packet-capture start *interface filename duration filesize* [*ring-buffer-number*] ["*filter* "]

Parameters

interface
: Network interface name, such as **eth0**

filename
: Output trace file name. If *ring-buffer-number* is also specified, an index number is appended to the filename.

duration
: Packet capture duration, in seconds. A value of **0** indicates the packet capture should continue until the packet-capture stop command is issued.

filesize
: Size of trace file, in megabytes (MB). The value must be from 0 through 100.

ring-buffer-number
: If specified, enables ring buffer function and sets number of ring buffer files. The value must be from 0 through 10.

filter
: Filter to use in selecting captured packets, within double quotation marks (\" \"). An extensive filter capability is supported, including, for example, filtering on packet source, destination, ports, and protocols. Search the internet for pcap-filter for complete details on supported filter syntax.

Usage Notes

- In order to start a packet capture on an interface, the following conditions are required:
  - The interface must be active.
  - No other packet captures can be active for the interface.
  - If tracing is required for an aggregate-interface, start the trace on the aggregate-interface itself. A packet capture on a member link may not contain complete data.
  - Any files from a previous packet capture must be cleared by using the packet-capture clear command.
- Packet capture stops when any of the following condition occurs:
  - The file size limitation is reached, if the ring-buffer function is not used.
  - The packet capture duration ends.
  - The packet-capture stop command is issued.
- Output trace files can be analyzed with commercial off-the-shelf tools that can read data in libpcap format, such as Wireshark.

Related Commands

See packet-capture clear, packet-capture list, and packet-capture stop.

Example

Start a packet capture on interface `eth1` for 60 seconds. Allow the trace files to grow to a

maximum of 10 MB. Write the packet capture to two ring buffer files named `pc11` and `pc12`.

```
Console> packet-capture start eth1 pc1 60 10 2 "icmp"
Packet capture started on eth1
Ok
Console>
```

Parent topic: Special purpose commands

packet-capture stop

Purpose

Stop an in-progress packet capture for a specified network interface.

Syntax

packet-capture stop *interface*

Parameters

interface
: Network interface for which packet capture needs to be stopped

Usage Notes

- Packet capture files remain in place until the packet-capture clear command is issued.
- Use file put to transfer the packet-capture files off the appliance for analysis.

Related Commands

See packet-capture clear, packet-capture list, and packet-capture start.

Example

Display active packet capture and then stop packet-capture activity on interface `eth1`.

```
Console> packet-capture list
interface       status          file(s)
eth1    Capturing               eth1pc1
                                eth1pc2
-------------------------------
eth0    Finished                eth0pc
-------------------------------
Ok
Console> packet-capture stop eth1
Stopping capture on eth1
Ok
Console>
```

Parent topic: Special purpose commands

platform collect-pd

Purpose

Collect a standard set of problem diagnosis data into a single file. This file can be transferred off the appliance for viewing or for sending to IBM support. The resulting output file is the output of a predefined list of command line interface (CLI) commands. The intention of this command is to "script" a number of command line interface (CLI) commands a support person might ask the customer to run as part of problem diagnosis.

Syntax

platform collect-pd [*PDfilename*]

Parameters

PDfilename
: Specifies the name of the text file to which the output of the problem diagnostic commands is written. If not specified, the default value is collect-pd.txt. If the output file exists, it is overwritten.

Usage Notes

- The output might include both clear text that customers can read, and obscured textual data intended only for IBM support.
- The output of this command is also included automatically in the data collected by the platform must-gather command. So if users run platform must-gather, they do not need to separately run platform collect-pd.
- Because the output is all printable text, the contents of the output file might be printed inside the command line interface (CLI) session by using the show log command. This is helpful if the appliance loses all network connectivity.

Related Commands

See platform must-gather, file put, and show log.

Example

Collects diagnostic command data, sending the output to the default output file.

```
Console> platform collect-pd
Console> file list
collect-pd.txt 79336 bytes created Feb 6, 2012 9:32:17 PM
Console>
```

Parent topic: Special purpose commands

platform log-level get

This command is deprecated. Instead, use show logging-settings.
Parent topic: Special purpose commands

platform log-level set

This command is deprecated. Instead, use edit logging-settings.
Parent topic: Special purpose commands

platform service disable ssh

Purpose

- Disable an ssh platform service if it affects your security policies.

Syntax

platform service ssh ... platform service ssh disable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service enable ssh, platform service enable telnet, or platform service disable telnet.

Example

```
Console> help platform service
CWZBR02449I: platform service ssh ...
CWZBR02901I: platform service ssh disable
```

Parent topic: Special purpose commands

platform service disable telnet

Purpose

- Disable a telnet platform service security protocol.

Syntax

platform service telnet ... platform service telnet disable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service enable telnet, platform service enable ssh, or platform service disable ssh.

Example

```
Console> help platform service
CWZBR02449I: platform service telnet ...
CWZBR02901I: platform service telnet disable
```

Parent topic: Special purpose commands

platform service enable ssh

Purpose

- Enable an ssh platform service security protocol.

Syntax

platform service ssh ... platform service ssh enable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service disable ssh, platform service enable telnet, or platform service disable telnet.

Example

```
Console> help platform service
CWZBR02449I: platform service ssh ...
CWZBR02901I: platform service ssh enable
```

Parent topic: Special purpose commands

platform service enable telnet

Purpose

- Enable a telnet platform service security protocol.

Syntax

platform service telnet ... platform service telnet enable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service disable telnet, platform service enable ssh, or platform service disable ssh.

Example

```
Console> help platform service
CWZBR02449I: platform service telnet ...
CWZBR02901I: platform service telnet enable
```

Parent topic: Special purpose commands

platform must-gather

Purpose

Collect the must-gather diagnostic data required for reporting a problem to IBM support.

Syntax

platform must-gather *filename* [*PDfilename*]

Parameters

filename
: Output file containing all diagnostic data, in a compressed (.tgz) format

pdfilename
: Name of the text file to which the output of the problem diagnostic commands will be written. This file contains the same output collected by the `platform collect-pd` command. If not specified, the default value is `collect-pd.txt`. If the output file exists, the file is overwritten.

Usage Notes

- The output might include both clear text and obscured data intended to be read only by IBM support.

Related Commands

See platform collect-pd.

Example

Collect diagnostic data to a file called `problem-data.tgz`.

```
Console> platform must-gather problem-data.tgz
Console> file list
problem-data.tgz 3449276 bytes created Mar 27, 2012 11:15:48 AM
collect-pd.txt 572878 bytes created Mar 27, 2012 11:15:48 AM
Console>
```

Parent topic: Special purpose commands

raid delete

Purpose

Delete the configuration on the RAID card controller and also clean up volume and partition information to set the physical disk back to factory state.

Syntax

raid delete

Parameters

None.

Usage Notes

- The data on the disk will be lost.
- The configuration on the RAID card controller will be erased and will be automatically reconfigured at the next boot.
- The partition will be eliminated and the partition information will be erased on the disk.
- The state of all physical disks will be reset.
- Only volumes managed by the RAID card controller will be deleted.
- Unmanaged volumes will not be deleted.

Example

Delete configuration on the RAID controller.

```
Console> raid delete
Delete volume
Delete volume successfully
Console> status volume
No volume
Console> status raid physical
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 0
    Logical drive ID: 0
    Logical drive name: raid0
    State: unconfigured good drive
...
```

Parent topic: Special purpose commands

raid re-use

Purpose

Make a physical disk previously used in another RAID card controller available for use in the appliance. When a previously used disk is inserted into the appliance, it might be recognized by the RAID controller as a foreign disk. This command will make the disk usable by the RAID controller and automatically start the rebuilding process for the disk into the RAID array.

Syntax

raid re-use

Parameters

None.

Usage Notes

- The old data on the physical disk will be lost.
- This command will clear the foreign state of a used physical disk and also trigger the rebuilding process. After the building process completes, the physical disk will transition online.
- This command works for only one physical disk. If more than one foreign physical disk is inserted at a time, the command will not be able to make them into a RAID configuration.

Example

Clean used physical disk foreign state and start rebuilding.

```
Console> raid re-use
Bring physical drive back online after hot swap
Done
Console> status raid physical
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: rebuild
    Progress percent: 3
...
```

Parent topic: Special purpose commands

raid foreign-config import

Purpose

Importing a foreign configuration is supported on 9005. A foreign configuration is a RAID configuration that already exists on a replacement set of drives. Foreign configurations on drives can be cleared or imported into the appliances on which they are installed. By default, foreign configurations are cleared if they are found on a 9005 machine. Users may enable the importing of foreign configurations on their needs via CLI command or C++/Java API.

Syntax

raid foreign-config import

Parameters

None.

Usage Notes

To import the foreign configurations on the replacement disks, take the following steps in order:

1. Execute the command raid foreign-config import to enable the importing of foreign configurations.
2. Shutdown the appliance.
3. Replace its drives with the ones on which a foreign configuration already exists.

After powering on the appliance, execute status volume to see if the raid volume is online. And check if the foreign configuration is successfully imported. By default, the importing of the foreign configurations is disabled. When it is disabled, any foreign configurations found on the disks will be cleared. Data on the disks will be lost too when a RAID configuration is cleared.

Related Commands

See raid foreign-config clear.

Example

Import a foreign configuration on the RAID controller.

```
Console> raid foreign-config import
CWZBR02804I: Import of foreign configuration is enabled. Foreign configurations will be imported a
t reboot.
```

Parent topic: Special purpose commands

raid foreign-config clear

Purpose

Import of foreign configuration is disabled. Foreign configurations will be cleared at reboot. Importing a foreign configuration is supported on 9005. A foreign configuration is a RAID configuration that already exists on a replacement set of drives. Foreign configurations on drives can be cleared or imported into the appliances on which they are installed. By default, foreign configurations are cleared if they are found on a 9005 machine.

Syntax

raid foreign-config clear

Parameters

None.

Related Commands

See raid foreign-config import.

Example

Import a foreign configuration on the RAID controller is disabled.

```
Console> raid foreign-config clear
CWZBR02805I: Import of foreign configuration is disabled. Foreign configurations will be cleared a
t reboot.
```

Parent topic: Special purpose commands

request force-suspend

Purpose

Request an abrupt suspension of grid activities while maintaining a network connection to the grid. If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is still available.

Syntax

request force-suspend

Options

None.

Parameters

None.

Usage Notes

None.

Related Commands

See request resume and request suspend.

Example

```
Console> request force-suspend
Force Stopped all XC-10 processes
Forced recycle of appliance
[Mon Feb 25 2013 09:46:59]  request force-suspend completed
Console> start-progresss
CWZBR02115E: Unknown command "start-progress"
Console> start-progress
DNS 10.42.229.86 10.42.229.86
DNS 10.1.1.105 10.1.1.105
DNS 9.42.94.15 9.42.94.15
SUSPENDED
```

Parent topic: Special purpose commands

request resume

Purpose

If you have previously requested a suspension of grid activities either through a request suspend or a request force suspend command, then this command restores grid activities.

Syntax

request resume

Options

None.

Parameters

None.

Usage Notes

This command takes some time to completely restore activities. If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is still available.

Related Commands

See request suspend and request force-suspend.

Example

```
Console> request resume
[Mon Feb 25 2013 09:48:09]  request resume completed
Console> start-progress
DNS 10.42.229.86 10.42.229.86
DNS 10.1.1.105 10.1.1.105
DNS 9.42.94.15 9.42.94.15
Volume 1 mounted
Volume 2 mounted
STARTING (4% complete)
```

Parent topic: Special purpose commands

request suspend

Purpose

Request a graceful suspension of grid activities while maintaining a network connection to the grid. If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is still available.

Syntax

request suspend

Options

None.

Parameters

None.

Usage Notes

None.

Related Commands

See request resume and request force-suspend.

Example

```
Console> request suspend
Stopped all XC-10 processes
Recycle of appliance
[Mon Feb 25 2013 09:46:59]  request suspend completed
Console> start-progresss
CWZBR02115E: Unknown command "start-progress"
Console> start-progress
DNS 10.42.229.86 10.42.229.86
DNS 10.1.1.105 10.1.1.105
DNS 9.42.94.15 9.42.94.15
SUSPENDED
```

Parent topic: Special purpose commands

set-dns-search

Purpose

Configure domain name server search domains.

Syntax

set-dns-search [*domain* [... *domain*]]

Parameters

domain
: Domain name server search domains.

Usage Notes

- Invoking the set-dns-search command without specifying a domain clears the list.
- Each command invocation replaces the previously configured domains.

Related Commands

See get-dns-search, get-ntp-servers, and set-dns-servers.

Example

Configure and then display domain name server search domains.

```
Console> set-dns-search rtp.raleigh.ibm.com raleigh.ibm.com ibm.com
Ok
Console> get-dns-search
DNS search domains:
        rtp.raleigh.ibm.com
        raleigh.ibm.com
        ibm.com
Console>
```

Parent topic: Special purpose commands

set-dns-servers

Purpose

Configure IP addresses of one or more domain name servers.

Syntax

set-dns-servers [*server* [... *server*]]

Parameters

server
: IP address of the domain name server

Usage Notes

- Starting the set-dns-servers command without specifying a server clears the list.
- Each command invocation replaces the previously configured servers.

Related Commands

See get-dns-servers, get-dns-search, and set-dns-search.

Example

Configure and then display domain name servers.

```
Console> set-dns-servers 9.42.106.2  2002:92a:8f7a:106:9:42:106:42
Ok
Console> get-dns-servers
Domain (DNS) servers:
        9.42.106.2
        2002:92a:8f7a:106:9:42:106:42
Console>
```

Parent topic: Special purpose commands

set-ntp-servers

Purpose

Configure network time protocol servers.

Syntax

set-ntp-servers [*server* [... *server*]]

Parameters

server
: IP address or host name of the network time protocol server

Usage Notes

- Starting the set-ntp-servers command without specifying a server clears the list.
- Each command invocation replaces the previously configured servers.

Related Commands

See get-ntp-servers and timezone set.

Example

Configure and then display NTP servers.

```
Console> set-ntp-servers serverA.yourco.com 10.10.106.44
Ok
Console> get-ntp-servers
NTP servers:    serverA.yourco.com
        10.10.106.44
Console>
```

Parent topic: Special purpose commands

show commandref

Purpose

Display a reference list of all commands supported in the command line interface (CLI).

Syntax

show commandref

Parameters

None.

Usage Notes

- Output from this command is intended to be parsed by a program.
- Output might be lengthy, so use of the command from the serial console is not suggested.

Example

List commands.

```
Console> show commandref
<reference>

<commands>
  <entry>
    <name>alias</name>
    <brief>alias [name [value]]/brief>
    <detailed/>
  </entry>
......
```

Parent topic: Special purpose commands

show components

Purpose

Display component firmware versions on the appliance.

Syntax

show components

Parameters

None.

Usage Notes

None.

Example

Display firmware versions of various components.

```
Console> show components
BMC Firmware Version: 2.53
ServeRaid Firmware Version: 12.12.0-0039
```

Parent topic: Special purpose commands

show locales

Purpose

Display a list of language, country and charset values usable for the locale set command. The valid combinations of language and country are displayed, along with the default charset for that combination. Then the entire list of supported charsets is displayed.

Syntax

show locales

Parameters

None.

Usage Notes

Some languages have multiple dialects, for example Chinese (zh) has simplified (Hans) and traditional (Hant) dialects. To specify traditional Chinese, use language zh_Hant. Be sure to specify a charset that is supported by the client software that you will be using to communicate to the appliance.

Related Commands

None.

Example

List the locale values usable for the locale set command.

```
 Console> show locales
   CWZBR02866I: af
   CWZBR02866I: af NA UTF-8
   CWZBR02866I: af ZA UTF-8
   CWZBR02866I: agq
   CWZBR02866I: agq CM UTF-8
   ...
   CWZBR02866I: zh TW BIG5
   CWZBR02866I: zu
   CWZBR02866I: zu ZA UTF-8
   CWZBR02867I: supported charsets: Big5 EUC-JP EUC-KR GB18030 GB2312 IBM00858 IBM850 IBM857 IBM86
0
   IBM861 IBM862 IBM863 IBM864 IBM865 IBM866 IBM868 IBM869 ISO-2022-JP ISO-2022-JP-1 ISO-2022-JP-2
   ISO-2022-KR ISO-8859-1 ISO-8859-10 ISO-8859-13 ISO-8859-15 ISO-8859-2 ISO-8859-3 ISO-8859-4
   ISO-8859-5 ISO-8859-6 ISO-8859-7 ISO-8859-8 ISO-8859-9 KOI8-R KSC_5601 Shift_JIS US-ASCII UTF-1
6
   UTF-16BE UTF-16LE UTF-32 UTF-7 UTF-8 cp1363 cp851 macintosh x-mac-centraleurroman x-mac-cyrilli
c
   x-mac-greek x-mac-turkish
```

Parent topic: Special purpose commands

show log

Purpose

Display a list of logfiles or the contents of a log file.

Syntax

- Display the list of log files: show log list
- Display the contents of a log file: show log *filename*

Parameters

filename
: Name of file to be listed to console

Usage Notes

- For reporting problems to IBM Support, use the platform must-gather command.
- In the event that network problems make it impossible to retrieve the must-gather output, this command can be used to list the logs to the serial console. However, be aware that the serial port is slow and the log files might be very large. Listing them this way might take a long time.

Example

List the displayable log files and then list the contents of the default-log file.

```
Console> show log list
default-log
default-trace
Console> show log default-log
2012-02-17T19:25:43+00:00 CWZBR00028 [info][firmwaremgr(3929)]: The Firmware Manager has started.
2012-02-17T19:25:43+00:00 CWZBR00031 [debug][firmwaremgr(3929)]: The Firmware Manager was created.
...
```

Parent topic: Special purpose commands

show timezones

Purpose

Display a list of time zones usable for the `timezone set` command.

Syntax

show timezones

Parameters

None.

Example

List the timezone values usable for the timezone set command.

```
Console> show timezones
EST5EDT
PST8PDT
...
```

Parent topic: Special purpose commands

show version

Purpose

Display the firmware version active on the appliance.

Syntax

show version

Parameters

None.

Example

Display the active firmware version.

```
Console> show version
Installation date: Mar 9, 2012 8:06:27 PM
Platform version: 4.9.4.0
Platform build ID: build12-20120309-1307
Platform build date: 2012-03-09 18:09:44+00:00
Machine type/model: 923572X
Serial number: 68A0553
Firmware type: Development
Console>
```

Parent topic: Special purpose commands

sshkey fingerprint

Purpose

Display the server keys used by the SSH server of the appliance.

Syntax

sshkey fingerprint

Parameters

None.

Example

Display SSH keys of the server.

```
Console> sshkey fingerprint
SSH RSA Key fingerprint 10:c6:c6:5c:44:72:d1:e9:84:15:df:65:47:96:1a:c1
SSH DSA Key fingerprint cd:a9:8e:a8:0b:d0:45:ce:ac:9e:67:92:06:cf:63:1b
Console>
```

Parent topic: Special purpose commands

sshkey reset

Purpose

Delete and regenerate the server keys used by the appliance's SSH server.

Syntax

sshkey reset

Parameters

None.

Related Commands

None.

Example

Reset the server's SSH keys.

```
Console> sshkey fingerprint
CWZBR02154I: SSH RSA Key fingerprint 10:c6:c6:5c:44:72:d1:e9:84:15:df:65:47:96:1a:c1
CWZBR02154I: SSH DSA Key fingerprint cd:a9:8e:a8:0b:d0:45:ce:ac:9e:67:92:06:cf:63:1b
Console> sshkey reset
CWZBR02196I: Ok
Console> sshkey fingerprint
CWZBR02154I: SSH RSA Key fingerprint 7c:43:c7:36:2e:4a:ed:2d:a9:af:62:cc:44:f0:a5:7b
CWZBR02154I: SSH DSA Key fingerprint c3:fb:08:aa:c8:81:81:57:f1:ff:9b:c3:70:df:4a:bd
Console>
```

Parent topic: Special purpose commands

sshkey verify

Purpose

Determine whether the server keys used by the appliance's SSH server are strong or weak.

Syntax

sshkey verify

Parameters

None.

Related Commands

None.

Example

Verify the strength of the server's SSH keys and reset them if they are not optimally strong.

```
Console> sshkey verify
CWZBR02797E: SSH keys are not optimally strong
Console> sshkey reset
CWZBR02196I: Ok
Console> sshkey verify
CWZBR02798I: Host SSH keys are strong
Console>
```

Parent topic: Special purpose commands

start-progress

Purpose

You can issue this command when the startup is in progress. The command displays the percentage of the startup process that has completed.

Syntax

start-progress
None.

Parameters

None.

Related Commands

None.

Example

```
Console> start-progress
DNS 9.42.139.134 9.42.139.134
DNS 9.42.139.139 9.42.139.139
DNS 9.42.139.152 xsa23e2.rtp.raleigh.ibm.com
DNS 9.42.139.142 localhost
CWXSA0014I: Volume 1 mounted
CWXSA0014I: Volume 2 mounted
CWXSA0006I: Catalog server started.
CWXSA0018I: Grid configuration service started.
CWXSA0017I: Container server 02 started.
CWXSA0017I: Container server 07 started.
CWXSA0017I: Container server 04 started.
CWXSA0017I: Container server 06 started.
CWXSA0017I: Container server 08 started.
CWXSA0017I: Container server 05 started.
CWXSA0017I: Container server 01 started.
CWXSA0017I: Container server 03 started.
CWXSA0002I: Grid administrative service started.
CWXSA0004I: Administrative console started.
STARTED
```

Parent topic: Special purpose commands

status battery

Purpose

Display expected battery expiration.

Syntax

status battery

Parameters

None.

Usage Notes

- The battery provides backup power for the hardware configuration and RAID disk writeable cache while the appliance is powered off and unplugged from an AC power source. Batteries are valid for a specified duration. This command displays the start and end of that duration.

Related Commands

See also status battery.

Example

Display expected battery expiration.

```
Console> status battery
Battery installed: Feb 17, 2012 7:25:45 PM
Battery expires: Feb 16, 2014 7:25:45 PM
Console>
```

Parent topic: Special purpose commands

status cpu-usage

This command is deprecated. Use status cpu-utilization instead.
Parent topic: Special purpose commands

status cpu-utilization

Purpose

Display CPU utilization over the past 10 seconds, 1 minute, 10 minutes, 1 hour, and 1 day intervals.

Syntax

status cpu-utilization

Parameters

None.

Usage Notes

None.

Example

Display CPU utilization.

```
Console> status cpu-utilization
CPU utilization over time:
0% over 10 seconds
49% over 1 minute
25% over 10 minutes
14% over 1 hour
6% over 1 day
Console>
```

Parent topic: Special purpose commands

status dynamic-tunnels

Purpose

Display details of currently active dynamic IPsec tunnels.

Syntax

status dynamic-tunnels *[tunnelID]*

Parameters

tunnelID
: Optional identifier of a specific dynamic tunnel to display in the form *Ynnn*. If a tunnel identifier is not specified, all of the existing dynamic IPsec tunnels are displayed.

Usage Notes

See "Network Security Configuration" for details about the fields returned by this status provider.

Example

Display the status of dynamic tunnel Y2.

```
Console> status dynamic-tunnels y2
Dynamic tunnel name: Y2
        AssociatedFiltCode: n/a
        AssociatedFiltCodeRange: n/a
        AssociatedFiltDestPort: n/a
        AssociatedFiltDestPortRange: n/a
        AssociatedFiltProtocol: ALL(0)
        AssociatedFiltSrcPort: n/a
        AssociatedFiltSrcPortRange: n/a
        AssociatedFiltType: n/a
        AssociatedFiltTypeRange: n/a
        AuthAlgorithm: HMAC-SHA1
        AuthInboundSpi: 4258054942 (0xFDCCC31E)
        AuthOutboundSpi: 796243132  (0x2F75B4BC)
        CurrentTime: 2012/06/01 15:27:14
        DiffieHellmanGroup: 14
        EncryptInboundSpi: 4258054942 (0xFDCCC31E)
        EncryptOutboundSpi: 796243132  (0x2F75B4BC)
        Generation: 1
        HowActivated: OnDemand
        HowToAuth: ESP
        HowToEncap: Transport
        HowToEncrypt: 3DES-CBC
        IKEVersion: 2.0
        InboundBytes: 192
        InboundPackets: 3
        IpFilterRule: tcpcs-ipsec-ifr
        KeyLength: n/a
        Lifesize: 0K
        LifesizeRefresh: 0K
        LifetimeExpires: 2012/06/01 19:17:54
        LifetimeRefresh: 2012/06/01 19:13:05
        LocalAddressBase: 9.42.90.6
        LocalAddressPrefix: n/a
        LocalAddressRange: n/a
```

```
LocalDynVpnRule: n/a
LocalEndPoint: 9.42.90.6
OutboundBytes: 192
OutboundPackets: 3
PFS: Yes
ParentIKETunnelID: K1
PendingNewActivation: n/a
RemoteAddressBase: 9.42.105.155
RemoteAddressPrefix: n/a
RemoteAddressRange: n/a
RemoteEndPoint: 9.42.105.155
RmtIpSpecExIDPayload: n/a
State: DONE
TunnelID: Y2
VpnActionName: tcpcs-idva
Console>
```

Parent topic: Special purpose commands

status fan

Purpose

Display fan speeds and status.

Syntax

status fan

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms, or blade-based platforms.

Example

Display fan status.

```
Console> status fan
Fan Speed Tray 1 Fan 1: 4800 (ok)
Fan Speed Tray 1 Fan 2: 5840 (ok)
Fan Speed Tray 1 Fan 3: 0.000000 (alert)
Fan Speed Tray 1 Fan 4: 6960 (ok)
Fan Speed Tray 2 Fan 1: 4720 (ok)
Fan Speed Tray 2 Fan 2: 5520 (ok)
Fan Speed Tray 2 Fan 3: 4720 (ok)
Fan Speed Tray 2 Fan 4: 6080 (ok)
Hard Disk Tray Fan 1: 4720 (ok)
Hard Disk Tray Fan 2: 6000 (ok)
Console>
```

Parent topic: Special purpose commands

status flash

Purpose

Display size and available space on the flash drive.

Syntax

status flash

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms.

Example

Display flash memory usage.

```
Console> status flash
Flash drive status:
        Total size = 3908 MB, free size = 1894 MB
Console>
```

Parent topic: Special purpose commands

status intrusion

Purpose

Display integrity status of physical appliance case.

Syntax

status intrusion

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms, or blade-based platforms.

Related Commands

See device intrusion clear.

Example

Display case integrity status.

```
Console> status intrusion
Case has not been opened and is secure
Console>
```

Parent topic: Special purpose commands

status memory

Purpose

Display RAM memory usage.

Syntax

status memory

Parameters

None.

Example

Display memory usage.

```
Console> status memory
Memory size 8193636K, free 7696564K
Console>
```

Parent topic: Special purpose commands

status nvdimm

Purpose

Display nvDIMM state.

Syntax

status nvdimm

Parameters

None.

Example

Display nvDIMM status.

```
Console> status nvdimm
  CWZBR02911I: All nvDIMMs are ready to go
  Console>
```

Parent topic: Special purpose commands

status power-supply

Purpose

Display status of power supply units.

Syntax

status power-supply

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms, or blade-based platforms.

Example

Display status of power supply units.

```
Console> status power-supply
Power Supply 1 Output Failure: true (alert)
Power Supply 2 Output Failure: false (ok)
Console>
```

Parent topic: Special purpose commands

status raid all

Purpose

Display details of physical disks and logical disks that are controlled by the RAID card controller.

Syntax

status raid all

Parameters

None.

Usage Notes

- Logical disk information here is not supported yet, so only physical disk detail is displayed.

Related Commands

See status volume and status raid physical.

Example

Show all physical disks and logical disks controlled by the RAID card controller.

```
Console> status raid all
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
    Interface type: SAS
    Interface speed: 6.0Gb/s
    SAS address: 5000c500286968d50000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE2CE1F
Position: HDD0
    Controller ID: 1
    Device ID: 9
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
    Interface type: SAS
```

```
    Interface speed: 6.0Gb/s
    SAS address: 5000c500286970090000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE28HV9
Console>
```

Parent topic: Special purpose commands

status raid battery

Purpose

Display the RAID BBU information.

Syntax

status raid battery

Parameters

None.

Usage notes

The BBU information is available only on appliances with MegaRAID controller, but not with MPT controller.

Example

Display the RAID BBU information.

```
Console> status raid battery
 CWZBR02808I: BBU status for Adapter '0':
 CWZBR02809I:   Battery Type                                : iBBU
 CWZBR02810I:   Voltage                                     : 4055 mV
 CWZBR02811I:   Current                                     : 0 mA
 CWZBR02812I:   Temperature                                 : 29 C
 CWZBR02816I: BBU Firmware Status:
 CWZBR02819I:   Charging Status                             : None
 CWZBR02820I:   Voltage                                     : OK
 CWZBR02821I:   Temperature                                 : OK
 CWZBR02822I:   Learn Cycle Requested                       : No
 CWZBR02823I:   Learn Cycle Active                          : No
 CWZBR02824I:   Learn Cycle Status                          : OK
 CWZBR02825I:   Learn Cycle Timeout                         : No
 CWZBR02826I:   I2c Errors Detected                         : No
 CWZBR02827I:   Battery Pack Missing                        : No
 CWZBR02828I:   Battery Replacement required                : No
 CWZBR02829I:   Remaining Capacity Low                      : No
 CWZBR02830I:   Periodic Learn Required                     : No
 CWZBR02831I:   Transparent Learn                           : No
 CWZBR02832I:   No space to cache offload                   : No
 CWZBR02833I:   Pack is about to fail. Should be replaced   : No
 CWZBR02834I:   Cache Offload premium feature required      : No
 CWZBR02835I:   Module microcode update required            : No
 CWZBR02847I: BBU Capacity Info:
 CWZBR02848I:   Relative State of Charge                    : 98 %
 CWZBR02849I:   Absolute State of Charge                    : 96 %
 CWZBR02850I:   Remaining Capacity                          : 1163 mAh
 CWZBR02851I:   Full Charge Capacity                        : 1187 mAh
 CWZBR02853I:   Run time to empty (min)                     : Battery is not being discharged
 CWZBR02855I:   Average time to empty (min)                 : Battery is not being discharged
 CWZBR02857I:   Average Time to full (min)                  : Battery is not being charged
 CWZBR02858I:   Cycle Count                                 : 21
 CWZBR02859I:   Max Error                                   : 6 %
 CWZBR02860I:   Remaining Capacity Alarm                    : 120 mAh
 CWZBR02861I:   Remaining Time Alarm                        : 10 Min
 CWZBR02836I: BBU Design Info:
```

```
CWZBR02837I:   Device Name                          : 3150301
CWZBR02838I:   Serial Number                        : 4010
CWZBR02839I:   Manufacturer Name                    : LS1121001A
CWZBR02840I:   Date of Manufacture                  : 07-03-2010
CWZBR02841I:   Design Capacity                      : 1215 mAh
CWZBR02843I:   Design Voltage                       : 3700 mV
CWZBR02844I:   Transparent Learn                    : 0
CWZBR02845I:   Specification Info                   : 33
CWZBR02846I:   Pack Stat Configuration              : 0x6490
Console>
```

Parent topic: Special purpose commands

status raid physical

Purpose

Display details of physical disks that are controlled by the RAID card controller.

Syntax

status raid physical

Parameters

None.

Usage Notes

- Detailed information displayed will be different based on different RAID card controllers.
- The *Location* or *Position* information corresponds with the physical location of the disks on the appliance. On the hard drive tray or front panel of the appliance, text is provided with this location information.

Related Commands

See status volume and status raid all.

Example

Show all physical disks controlled by the RAID card controller.

```
Console> status raid physical
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
    Interface type: SAS
    Interface speed: 6.0Gb/s
    SAS address: 5000c500286968d500000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE2CE1F
Position: HDD0
    Controller ID: 1
    Device ID: 9
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
```

```
    Interface type: SAS
    Interface speed: 6.0Gb/s
    SAS address: 5000c5002869700900000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE28HV9
Console>
```

Parent topic: Special purpose commands

status voltage

Purpose

Display voltage readings and status.

Syntax

status voltage

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms.

Example

Display voltage readings.

```
Console> status voltage
Voltage +12: 11.8 (ok)
Voltage +5 Standby: 4.90 (ok)
Voltage +5: 4.87 (ok)
Voltage +3.3: 3.33 (ok)
Voltage +1.8: 1.79 (ok)
Voltage +1.5: 1.49 (ok)
Voltage CPU1 Core: 1.07 (ok)
Voltage CPU2 Core: 1.08 (ok)
Voltage Bus Termination: 1.08 (ok)
Voltage Battery: 3.14 (ok)
Console>
```

Parent topic: Special purpose commands

status volume

Purpose

Display all volume and partition information about the system.

Syntax

status volume

Parameters

None.

Usage Notes

- This command shows both managed and unmanaged types of volumes and partitions.
- Unmanaged partitions does not show the encryption information.
- Unmanaged partitions is shown as offline if the mount point does not exist.

Related Commands

See status raid physical and status raid all.

Example

Show all volume details.

```
Console> status volume
Storage Volume Status:
Volume 'raid0' is [Online] : size = 285148 MB, num partitions = 2
    Partition 1 'raid0-1' is [Online], encryption is [None]
        Total size = 91 MB,   free size = 86 MB
    Partition 2 'raid0-2' is [Online], encryption is [None]
        Total size = 184 MB,   free size = 179 MB
Console>
```

Parent topic: Special purpose commands

status temperature

Purpose

Display temperature status of physical hardware components.

Syntax

status temperature

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- This command is not applicable to virtual platforms.

Example

Display temperature status.

```
Console> status temperature
Power Supply 1 Over-Temperature: false (ok)
Power Supply 1 High Temperature: false (ok)
Power Supply 2 Over-Temperature: false (ok)
Power Supply 2 High Temperature: false (ok)
Temperature System 1: 30.0 (ok)
Temperature System 2: 25.0 (ok)
Temperature CPU1: 27.0 (ok)
Temperature CPU2: 30.0 (ok)
Temperature Memory 00: 45.0 (ok)
Temperature Memory 10: 41.0 (ok)
Temperature Memory 20: 39.0 (ok)
Temperature Memory 30: 39.0 (ok)
Temperature Ethernet Ambient: 29 (ok)
Temperature Ethernet MAC: 50 (ok)
Console>
```

Parent topic: Special purpose commands

status uptime

Purpose

Display the length of time since the appliance was booted.

Syntax

status uptime

Parameters

None.

Example

Display length of time appliance has been up.

```
Console> status uptime
13 days 17 hours 13 minutes 15 seconds
Console>
```

Parent topic: Special purpose commands

timezone get

Purpose

Display the configured timezone.

Syntax

timezone get

Parameters

None.

Usage Notes

- None.

Related Commands

See timezone set, datetime get, and datetime set.

Example

Display the configured time zone.

```
Console> timezone get
Timezone is EST
Console>
```

Parent topic: Special purpose commands

timezone set

Purpose

Set the timezone for the local time.

Syntax

timezone set *zone*

Parameters

zone
: Timezone value

Usage Notes

- None.

Related Commands

See timezone get, datetime get, and datetime set.

Example

Set the timezone to Eastern Standard Time (EST) and confirm the setting.

```
Console> timezone set EST
Console> timezone get
Timezone is EST
Console>
```

Parent topic: Special purpose commands

unalias

Purpose

Remove a configured command line interface (CLI) command alias.

Syntax

unalias *name*

Parameters

name
: Alternate name for command line interface (CLI) command that is to be removed

Related Commands

See alias.

Example

Show configured aliases and then remove the alias called `test-connectivity`.

```
Console> alias
alias test-connectivity "net-test ping 9.42.106.2"
Console> unalias test-connectivity
Console> alias
No aliases defined
Console>
```

Parent topic: Special purpose commands

usage

Purpose

Display details about how to use the config command.

Syntax

```
config usage
```

Parameters

None

Usage notes

None

Example

Display details about the config command.

```
config usage
```

Parent topic: Special purpose commands

user add

Purpose

Define a new user to the appliance.

Syntax

user add *username* [*defaultpass*]

Parameters

username
: User name to be added. User names must consist of alphanumeric characters or the special characters: dot(.), dash(-), underscore(_) or plus(+).

defaultpass
: Default password to assign to new user name

Usage Notes

- If the password is not provided on the initial command invocation, the command line interface (CLI) prompts for the password and a confirmation of the password.

Related Commands

See user list, user delete, user password.

Example

Add users `fred` and `barney`.

```
Console> user add fred
enter default password:**********
confirm password:**********
Ok
Console> user add barney rubble
Ok
Console> user list
User names:
     admin
     barney
     fred
Console>
```

Parent topic: Special purpose commands

user delete

Purpose

Delete a user from the appliance.

Syntax

user delete *username*

Parameters

username
: User name to be deleted

Related Commands

See also user add and user list.

Example

Delete user `fred`.

```
Console> user list
User names:
        admin
        fred

Console> user delete fred
Ok
Console> user list
User names:
        admin
Console>
```

Parent topic: Special purpose commands

user known-hosts list

Purpose

List the known hosts for the current user. When the user issues command file get|put scp://user@host/dir, the public key of the host is saved in a local file. Use this command to list the hosts that are saved in that file.

Syntax

user known-hosts list

Parameters

None.

Related Commands

Example

List the known hosts for the currently logged-in user.

```
Console> user known-hosts list
  aaa.example.com,192.168.10.200
  bbb.example.com,192.168.11.201
  Console>
```

Parent topic: Special purpose commands

user known-hosts delete

Purpose

Delete the stored public key for the named host from the known hosts file for the current user. When the user issues command file get|put scp://user@host/dir, the public key of the host is saved in a local file, and used to verify the host's identity on subsequent file get|put scp:// commands. If the host changes its key, that verification can fail. Use this command to delete the public key stored for a known host, so the host's new key can be learned.

Syntax

user known-hosts delete hostname

Parameters

hostname
Host name or IP address of the known hosts entry to be deleted.

Related Commands

See user known-hosts list

Example

Delete the public key for known host aaa.example.com from the currently logged-in user's known hosts list.

```
<p>Console&gt; user known-hosts delete aaa.example.com
  CWZBR02196I: Ok
  Console&gt;</p>
```

Parent topic: Special purpose commands

user list

Purpose

List users defined on the appliance.

Syntax

user list

Parameters

None.

Related Commands

See user add and user delete.

Example

List currently defined users.

```
Console> user list
User names:
        admin
        fred
Console>
```

Parent topic: Special purpose commands

user password

Purpose

Change the password for the currently active user.

Syntax

user password *oldpass newpass*

Parameters

oldpass
: Old (currently active) password

newpass
: New password

Usage Notes

- If the old or new password is not provided on the initial command invocation, the command line interface (CLI) prompts for the appropriate passwords and a confirmation of the new password.

Related Commands

See user add.

Example

Change password three times.

```
Console> user password aaaa bbbb
Changing password...
Ok
Console> user password bbbb
new password:****
confirm password:****
Changing password...
Ok
Console> user password
current password:****
new password:*****
confirm password:*****
Changing password...
Ok
Console>
```

Parent topic: Special purpose commands

user sshkey add

Purpose

Retrieve an SSH public key file from the specified URL for use by the current user.

Syntax

user sshkey add *url*

Parameters

url
: URL of public key file

Usage Notes

- Once the key file is stored, an ssh client with the corresponding private key can ssh in as that user without specifying a password.
- Only one public key is supported per appliance user. Subsequent user sshkey add commands replace the public key file for that appliance user.

Related Commands

See user sshkey delete.

Example

Retrieve a public key for the currently logged-in user.

```
Console> user sshkey add scp://user1@system1.rtp.raleigh.ibm.com:~/mypublickey
user1@system1.rtp.raleigh.ibm.com's password:
mypublickey                                   100%  411     0.4KB/s   00:00
Wrote 411 bytes to local storage
Adding ssh public key file for user wilma
Ok
Console>
```

Parent topic: Special purpose commands

user sshkey delete

Purpose

Delete the stored SSH public key file for the current user.

Syntax

user sshkey delete

Parameters

None.

Related Commands

See user sshkey add.

Example

Delete the public key for the currently logged-in user.

```
Console> user sshkey delete
Deleting ssh public key file for user wilma
Ok
Console>
```

Parent topic: Special purpose commands

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

Packages	
com.ibm.websphere.objectgrid	These are the main application APIs for the ObjectGrid.
com.ibm.websphere.objectgrid.client	This package contains the primary interfaces and classes for setting behaviors for ObjectGrid clients for this process.
com.ibm.websphere.objectgrid.config	This package contains the interfaces and a factory class for creating ObjectGrid configuration objects programatically.
com.ibm.websphere.objectgrid.continuousquery	
com.ibm.websphere.objectgrid.continuousquery.exception	
com.ibm.websphere.objectgrid.continuousquery.filter	These are the Filter implementations which are used to build continuous query definitions.
com.ibm.websphere.objectgrid.management	This package contains the interfaces for all ObjectGrid MBeans.
com.ibm.websphere.objectgrid.plugins	These are the interfaces for adding plugins to the Grid core framework.
com.ibm.websphere.objectgrid.plugins.builtins	This package contains built-in plugins for ObjectGrid core.
com.ibm.websphere.objectgrid.plugins.index	
com.ibm.websphere.objectgrid.plugins.io.annotations	
com.ibm.websphere.objectgrid.	This package has the class MapPermission and class AdminPermission which represents the permissions for to access the ObjectGrid maps and

security	ObjectGrid administration respectively.
com.ibm.websphere.objectgrid.security.config	This package contains the ObjectGrid client security configurations.
com.ibm.websphere.objectgrid.security.plugins	This package contains the interfaces for adding plug-ins to the ObjectGrid security framework and assoicated Exception classes.
com.ibm.websphere.objectgrid.security.plugins.builtins	This package contains the built-in implementation for the security plugins.
com.ibm.websphere.objectgrid.spring	This package holds the Spring specific APIs for ObjectGrid.
com.ibm.websphere.xs.ra	These are the eXtreme Scale Resource Adapter APIs that allows integration with a Java EE application.
com.ibm.websphere.xsa	

Package com.ibm.websphere.objectgrid

These are the main application APIs for the ObjectGrid.

See:

Description

Interface Summary

BackingMap	This is the public interface to the BackingMap.
CatalogDomainInfo	Identifies the configuration attributes of a catalog service domain.
CatalogDomainManager	Provides access to catalog domain configuration information for the current environment.
ClientClusterContext	This interface is a context to represent which cluster/domain the client connected to using one of the `ObjectGridManager.connect` methods.
ClientReplicableMap	**Deprecated.** *The client replicated map function is deprecated in version 8.6.*
IObjectGridException	This interface is used to ensure JDK 1.4 Throwable chaining behavior for all exceptions thrown by ObjectGrid even when an earlier JDK is used (e.g.
JavaMap	This interface is a handle to a named Map.
ObjectGrid	This object is used for creating sessions to the ObjectGrid.
ObjectGridManager	ObjectGridManager is responsible for creating or retrieving local ObjectGrid instances and connecting to distributed ObjectGrid servers.
ObjectMap	This is a handle to a named Map.
PartitionManager	This interface will be used for calculating the proper partition for a given input key.
Session	This interface represents a session container for `ObjectMaps`.
StateManager	The StateManager can be used to retrieve the availability state of an ObjectGrid.
TxID	This interface is an opaque identifier for a transaction.

Class Summary

AvailabilityState	Each shard in a distributed ObjectGrid has an availability state associated with it.
ClientReplicableMap.Mode	Client Replication mode
CopyMode	This class is used to define the "copy" mode when the `setCopyMode` method of the `BackingMap` interface is used.
LockStrate	LockStrategy provides an enumerated type idiom for use on the

gy	`BackingMap.setLockStrategy(LockStrategy)` method.
ObjectGridManagerFactory	This factory class is a high level helper class to get `ObjectGridManager` instances.
TTLType	Every BackingMap in ObjectGrid has a built in timed based evictor that is referred to as "time to live" evictor or TTL evictor.

Enum Summary

ObjectMap.PutMode	Identifies the operation mode of the `ObjectMap.put(Object, Object)`, `ObjectMap.putAll(Map)` , `JavaMap.put(Object, Object)` and `JavaMap.putAll(Map)` methods.
Session.TxCommitProtocol	The commit protocols that can be used to commit the Session's transaction

Exception Summary

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException	This exception is thrown when a method call to the Client/Server TransactionCallback detects the user is attempting to perform a write against multiple maps in different Map Sets, Partition Sets or Replication groups.
ConnectException	This exception is used to indicate that the client was unable to connect to the server
DuplicateKeyException	A DuplicateKeyException exception is thrown if a key cannot be inserted into a BackingMap because an object with the same key already exists.
KeyNotFoundException	Normally, record not found means a null is returned.
LockDeadlockException	This exception is used by the lock manager to indicate that it detected a deadlock.
LockException	A general locking exception indicating something went wrong with locking operations.
LockInternalFailureException	This exception is used by the lock manager to indicate it detected some internal programming error while processing a lock or unlock request.
LockTimeoutException	This exception is used by the lock manager to indicate that the maximum wait time for a lock has been exceeded.
NoActiveTransactionException	An exception indicating there is no active transaction.
ObjectGridException	Base exception class for all checked exceptions thrown by the ObjectGrid product.
ObjectGrid	

RuntimeException	This exception is the base class for all runtime exceptions thrown by the cache.
ReadOnlyException	This exception is thrown when an attempt is made to modifying operations on a read only maps.
ReplicationVotedToRollbackTransactionException	This exception is thrown when a transaction was rolled back because some/all of the replicas failed to apply the transaction when in synchronous replication mode.
SessionNotReentrantException	A Session object can only be used by a single thread concurrently to perform map operations.
TargetNotAvailableException	A `TargetNotAvailableException` indicates the ObjectGrid target is not available.
TransactionAffinityException	This exception is thrown for inflight transaction when server fails over.
TransactionAlreadyActiveException	An exception indicating a transaction is already active for the current session.
TransactionException	A general transaction exception indicating something went wrong with a transaction.
TransactionQuiesceException	This exception is thrown when partition/shard/mapset/replication group/ replication group member/server/cluster/objectgrid is entered quiesce process for various reasons such as shard movement, partition relocation, system update, server shutdown, and others.
TransactionTimeoutException	This exception is thrown when a transaction exceeds the transaction timeout that was specified on the `ObjectGrid` or `Session`.
UnavailableServiceException	This exception is thrown when all servers are dead or when all services are unavailable even though servers are running.
UndefinedMapException	This exception indicates that the map which an application tries to access is not defined in the ObjectGrid.

Package com.ibm.websphere.objectgrid Description

These are the main application APIs for the ObjectGrid. The main interface here is the ObjectGrid interface. A JVM needs to create at least one instance.

Introduction

The WebSphere ObjectGrid is designed as a data caching tier that can be used to hold data from multiple sources and then make it available to the clients of the ObjectGrid. The clients access the data through the ObjectGrid APIs. The ObjectGrid is designed to be able to store large quantities of data.

Programming Tutorial

The following sections show snippets on the usage of the ObjectGrid APIs.

Obtaining a ObjectGrid instance.

The application needs to construct an ObjectGrid reference first. An application can choose to make several ObjectGrid instances. Each instance is independent, however, and has it's own configuration file. For now, use the following code and programmatically initialize it using the setter methods on the ObjectGrid.

```
ObjectGrid objectGrid = new ObjectGridImpl();
```

The instance can then have a Map defined on it using the following snippet:

```
BackingMap bm = objectGrid.defineMap("TABLE1");
```

Again, setter methods on BackingMap allow it to be configured once it's defined.

Working with an ObjectGrid, Sessions

Each thread that wants to access the ObjectGrid must have its own Session instance. The ObjectGrid class has a getSession method that returns one. Once the thread has a Session then it can obtain ObjectMap instances for manipulating data in the ObjectGrid as well as use the begin/commit/rollback methods on the Session to handle transactions.

```
Session session = objectGrid.getSession();
ObjectMap table1 = session.getMap("TABLE1");
session.begin();
MyData d = (MyData)table1.get("key1");
session.commit();
```

Package com.ibm.websphere.objectgrid.client

This package contains the primary interfaces and classes for setting behaviors for ObjectGrid clients for this process.

See:

Description

Interface Summary	
ClientProperties	The set of properties used to define various preference for ObjectGrid clients.

Package com.ibm.websphere.objectgrid.client Description

This package contains the primary interfaces and classes for setting behaviors for ObjectGrid clients for this process.

Overview

The interfaces in this package should not be implemented directly but are used by the `ClientClusterContext` to set default behaviors for application clients for an ObjectGrid instance.

The properties available for use are defined in the `ClientProperties` interface.

There are two ways to configure client properties:

1. Create a properties file named `objectGridClient.properties` and store it in the root of your classpath.
2. Create a properties file on your file system in the directory where the client is started from named objectGridClient.properties.
3. Create a properties file with any path and name and use the following system property to detect it: `-Dcom.ibm.websphere.objectgrid.ClientProperties=<fileName>`
4. Create a properties file with any path and name and set load it programmatically and pass url to ClientClusterContext.getClientProperties(ogname, url).
5. Programmatically define the properties the `ClientProperties` set methods.

In the following example we set the proximity routing defaults for all clients that use this ClientClusterContext:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ClientClusterContext ccc = ogMgr.connect(...);
ClientProperties props = ccc.getClientProperties("myOGName");
props.setPreferLocalHost(true);
props.setPreferLocalProcess(true);
props.setPreferZones(new String[]{"New York", "Texas"});

// The ClientProperites are now applied to the ObjectGrid client connection:
ObjectGrid og=ogMgr.get(ccc, "myOGName");
```

The following example uses a custom client properties file:

```
ClientClusterContext ccc = ogMgr.connect(...);
```

```
    URL clientPropsURL = Thread.currentThread().getContextClassLoader().getResource("etc/myObjectG
ridClient.properties");
    ClientProperties props = ccc.setClientProperties("myOGName", clientPropsURL);

    // The ClientProperites are now applied to the ObjectGrid client connection:
    ObjectGrid og=ogMgr.get(ccc, "myOGName");
```

The following file is an example of a properties file that matches the proceeding API:

```
    preferLocalProcess=true
    preferLocalhost=true
    preferZones=New York,Texas
```

---

com.ibm.websphere.objectgrid.client

Interface ClientProperties

public interface **ClientProperties**

The set of properties used to define various preference for ObjectGrid clients.

See the package summary for details on how to use the ClientProperties class and properties file.

Since:
WAS XD 6.1.0.3, XC10

Field Summary

static String	**CLIENT_PROPS_FILE_PATH_KEY** The system property key to override the location of the client properties file.
static String	**DEFAULTCLIENTPROPERTYFILE** The default name of client property file
static String	**PROP_LISTENER_HOST** Listener host property key for the client properties file.
static St	**PROP_LISTENER_PORT** Listener port property key for the client properties file.

String	
static String	**PROP_MAXIMUM_XIO_NETWORK_THREAD_POOL_SIZE** Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.
static String	**PROP_MAXIMUM_XIO_WORKER_THREAD_POOL_SIZE** Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.
static String	**PROP_MINIMUM_XIO_NETWORK_THREAD_POOL_SIZE** Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.
static String	**PROP_MINIMUM_XIO_WORKER_THREAD_POOL_SIZE** Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.
static String	**PROP_PREFER_LOCAL_HOST** Currently, this property is not used.
static String	**PROP_PREFER_LOCAL_PROCESS** Currently, this property is not used.

[ng](#)	
static [String](#)	**PROP_PREFER_ZONES** Prefer zones property key for the client properties file.
static [String](#)	**PROP_REQUEST_RETRY_TIMEOUT** The requestRetryTimeout which indicates how long to retry a request (in milliseconds).
static [String](#)	**PROP_SHUFFLE_BOOTSTRAP_ADDRESSES** The shuffleBoostrapAddresses property is used to determine if the catalog service grid addresses should be randomized when used by a client when bootstrapping to the grid.
static [String](#)	**PROP_XIO_REQUEST_TIMEOUT** The xioRequestTimeout indicates how long the eXtreme IO transport will wait for cross-process call to complete.
static [String](#)	**PROP_XIO_TIMEOUT** Sets the timeout for server requests using the eXtremeIO transport.

Method Summary

[String](#)	**getListenerHost**() Retrieves the host to be used by the ORB.

int	**getListenerPort**() Retrieves the port to be used by the ORB.
int	**getMaximumXIONetworkThreads**() Gets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.
int	**getMaximumXIOWorkerThreads**() Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.
int	**getMinimumXIONetworkThreads**() Gets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.
int	**getMinimumXIOWorkerThreads**() Retrieves the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.
String[]	**getPreferZones**() Retrieve the preferred zones.
long	**getRequestRetryTimeout**() Retrieves the current request retry timeout.
int	**getXIORequestTimeout**() Gets the current timeout that is allowed for requests to servers over the XIO transport to complete.
int	**getXIOTimeout**() Returns the current timeout for server requests using the XIO transport.
boolean	**isBootStrapListShuffled**() Retrieves the value of the BootStrapListShuffled property.
boolean	**isPreferLocalHost**() This method is reserved for future use.
boolean	**isPreferLocalProcess**() This method is reserved for future use.
void	**setBootStrapListShuffled**(boolean shuffle) Sets the value of the BootStrapListShuffled property.

void	**setMaximumXIONetworkThreads**(int maxThreads) Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.
void	**setMaximumXIOWorkerThreads**(int maxThreads) Retrieves the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.
void	**setMinimumXIONetworkThreads**(int minThreads) Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.
void	**setMinimumXIOWorkerThreads**(int minThreads) Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.
void	**setPreferLocalHost**(boolean localHost) This method is reserved for future use.
void	**setPreferLocalProcess**(boolean localProcess) This method is reserved for future use.
void	**setPreferZones**(String[] zones) Prefer routing to specific zones.
void	**setRequestRetryTimeout**(long requestRetryTimeout) Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions.
void	**setXIORequestTimeout**(int timeout) Sets the current timeout that is allowed for requests to servers over the XIO transport to complete.
void	**setXIOTimeout**(int timeout) Sets the current timeout for server requests using the XIO transport.

Field Detail

DEFAULTCLIENTPROPERTYFILE

static final String **DEFAULTCLIENTPROPERTYFILE**

The default name of client property file

See Also:
Constant Field Values

CLIENT_PROPS_FILE_PATH_KEY

static final String **CLIENT_PROPS_FILE_PATH_KEY**

The system property key to override the location of the client properties file.

Since:
7.0
See Also:
Constant Field Values

PROP_PREFER_LOCAL_PROCESS

`static final String` **`PROP_PREFER_LOCAL_PROCESS`**

Currently, this property is not used. It is reserved for future use.

See Also:
`setPreferLocalProcess(boolean)`, Constant Field Values

PROP_PREFER_LOCAL_HOST

`static final String` **`PROP_PREFER_LOCAL_HOST`**

Currently, this property is not used. It is reserved for future use.

See Also:
`setPreferLocalHost(boolean)`, Constant Field Values

PROP_PREFER_ZONES

`static final String` **`PROP_PREFER_ZONES`**

Prefer zones property key for the client properties file. Each specified zone is separated by a comma in the form: preferZones=ZoneA,ZoneB,ZoneC

See Also:
`setPreferZones(String[])`, Constant Field Values

PROP_REQUEST_RETRY_TIMEOUT

`static final String` **`PROP_REQUEST_RETRY_TIMEOUT`**

The requestRetryTimeout which indicates how long to retry a request (in milliseconds). A 0 indicates that the request should fail fast and skip over in internal retry logic. Exceptions that cannot succeed even if tried again such as DuplicateException will be returned immediately.

Since:
7.0
See Also:
`setRequestRetryTimeout(long)`, Constant Field Values

PROP_XIO_REQUEST_TIMEOUT

`static final String` **`PROP_XIO_REQUEST_TIMEOUT`**

The xioRequestTimeout indicates how long the eXtreme IO transport will wait for cross-process call to complete. The value is expressed in milliseconds. The default value is 30,000 or 30 seconds. When custom tuning client side retry of eXtreme Scale operations, this value will determine how long a single network operation is given, and then the

PROP_REQUEST_RETRY_TIMEOUT / requestRetryTimeout will control how long those individual network operations are retried.

Since:
8.6.0.2, XC10 2.5
See Also:
Constant Field Values

PROP_LISTENER_HOST

`static final String` **`PROP_LISTENER_HOST`**

Listener host property key for the client properties file.

Since:
XS 7.1
See Also:
`getListenerHost()`, Constant Field Values

PROP_LISTENER_PORT

`static final String` **`PROP_LISTENER_PORT`**

Listener port property key for the client properties file.

Since:
XS 7.1
See Also:
`getListenerPort()`, Constant Field Values

PROP_SHUFFLE_BOOTSTRAP_ADDRESSES

`static final String` **`PROP_SHUFFLE_BOOTSTRAP_ADDRESSES`**

The shuffleBoostrapAddresses property is used to determine if the catalog service grid addresses should be randomized when used by a client when bootstrapping to the grid. The default value of the property is true.

Since:
7.1.0.3
See Also:
Constant Field Values

PROP_XIO_TIMEOUT

`static final String` **`PROP_XIO_TIMEOUT`**

Sets the timeout for server requests using the eXtremeIO transport. The timeout is set in seconds. The default xioTimeout for server requests is set to 30 seconds. The valid range is timeout >= 1.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MAXIMUM_XIO_NETWORK_THREAD_POOL_SIZE

static final String **PROP_MAXIMUM_XIO_NETWORK_THREAD_POOL_SIZE**

Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MAXIMUM_XIO_WORKER_THREAD_POOL_SIZE

static final String **PROP_MAXIMUM_XIO_WORKER_THREAD_POOL_SIZE**

Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MINIMUM_XIO_NETWORK_THREAD_POOL_SIZE

static final String **PROP_MINIMUM_XIO_NETWORK_THREAD_POOL_SIZE**

Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MINIMUM_XIO_WORKER_THREAD_POOL_SIZE

static final String **PROP_MINIMUM_XIO_WORKER_THREAD_POOL_SIZE**

Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

Method Detail

setPreferZones

void **setPreferZones**(String[] zones)

Prefer routing to specific zones.

When zones are enabled on an ObjectGrid, requests will be routed to the specified zones.

Parameters:
`zones` - array of zone names. If null or an empty array, then requests are routed to all zones.

setPreferLocalProcess

```
void setPreferLocalProcess(boolean localProcess)
```

This method is reserved for future use. Calls to the method will not result in any performed operation.

Parameters:

`localProcess` -

---

setPreferLocalHost

```
void setPreferLocalHost(boolean localHost)
```

This method is reserved for future use. Calls to the method will not result in any performed operation.

Parameters:

`localHost` -

---

getPreferZones

```
String[] getPreferZones()
```

Retrieve the preferred zones.

Returns:

the preferred zones.

---

isPreferLocalProcess

```
boolean isPreferLocalProcess()
```

This method is reserved for future use. The returned value should be ignored by the user.

Returns:

false

---

isPreferLocalHost

```
boolean isPreferLocalHost()
```

This method is reserved for future use. The returned value should be ignored by the user.

Returns:

false

---

setRequestRetryTimeout

```
void setRequestRetryTimeout(long requestRetryTimeout)
```

Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions. A request will timeout when either the request timeout expires or the transaction timeout expires, whichever expires first.

A value of 0 indicates that all requests should fail immediately and avoid any retry logic.

Exceptions that cannot succeed even if tried again such as DuplicateKeyException exceptions will be thrown immediately.

A value of -1 indicates that the request retry timeout is not set, meaning that the request duration is governed by the transaction timeout.

The request retry timeout can be overridden using the `Session.setRequestRetryTimeout(long)` method.

Parameters:

`requestRetryTimeout` - the duration in milliseconds retry a client request, 0 if the request should fail immediately or -1 if the request timeout is not set.

Since:

7.0

See Also:

`Session.setRequestRetryTimeout(long)`, `ObjectGrid.setTxTimeout(int)`

---

getRequestRetryTimeout

`long` **`getRequestRetryTimeout`**`()`

Retrieves the current request retry timeout. Returns -1 if it was not set.

Returns:

requestRetryTimeout in milliseconds, 0 to fail immediately or -1 if not set.

Since:

7.0

---

getListenerHost

`String` **`getListenerHost`**`()`

Retrieves the host to be used by the ORB. The listener host property defaults to 'localhost'. This property can only be set in the client.properties file.

Returns:

The host that the ORB will bind to.

Since:

7.1

---

getListenerPort

`int` **`getListenerPort`**`()`

Retrieves the port to be used by the ORB. The listener port property defaults to the corbaloc port, 2809. This property can only be set in the client.properties file.

Returns:

The port that the ORB will bind to.

Since:

7.1

---

isBootStrapListShuffled

`boolean` **`isBootStrapListShuffled`**`()`

Retrieves the value of the BootStrapListShuffled property.

Returns:
true if the value of BootStrapListeShuffled was set to true. false if the value of BootStrapListeShuffled was set to false.
Since:
7.1.0.3

---

setBootStrapListShuffled

`void` **`setBootStrapListShuffled`**`(boolean shuffle)`

Sets the value of the BootStrapListShuffled property.

Parameters:
`shuffle` - true the bootstrap list will be shuffled providing each client a random distribution of catalog servers to select from. false the first viable address in the list of catalog servers will be used.
Since:
7.1.0.3

---

getMinimumXIOWorkerThreads

`int` **`getMinimumXIOWorkerThreads`**`()`

Retrieves the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Returns:
the mininum number of threads.
Since:
8.6, XC10 2.5

---

setMinimumXIOWorkerThreads

`void` **`setMinimumXIOWorkerThreads`**`(int minThreads)`

Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Parameters:
`minThreads` - the minimum number of threads.
Since:
8.6, XC10 2.5

---

getMaximumXIOWorkerThreads

`int` **`getMaximumXIOWorkerThreads`**`()`

Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Returns:
the maximum number of threads.
Since:
8.6, XC10 2.5

---

setMaximumXIOWorkerThreads

`void` **`setMaximumXIOWorkerThreads`**`(int maxThreads)`

Retrieves the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Parameters:
`maxThreads` - the maximum number of threads.
Since:
8.6, XC10 2.5

---

getMinimumXIONetworkThreads

`int` **`getMinimumXIONetworkThreads`**`()`

Gets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Returns:
the minimum number of threads
Since:
8.6, XC10 2.5

---

setMinimumXIONetworkThreads

`void` **`setMinimumXIONetworkThreads`**`(int minThreads)`

Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Parameters:
`minThreads` - the minimum number of threads
Since:
8.6, XC10 2.5

---

getMaximumXIONetworkThreads

`int` **`getMaximumXIONetworkThreads`**`()`

Gets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Returns:
the maximum number of threads.
Since:
8.6, XC10 2.5

---

setMaximumXIONetworkThreads

`void` **`setMaximumXIONetworkThreads`**`(int maxThreads)`

Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Parameters:
`maxThreads` - the maximum number of threads.
Since:
8.6, XC10 2.5

---

getXIOTimeout

`int` **`getXIOTimeout`**`()`

Returns the current timeout for server requests using the XIO transport.

Returns:
the current timeout in seconds
Since:
8.6, XC10 2.5

---

setXIOTimeout

`void` **`setXIOTimeout`**`(int timeout)`

Sets the current timeout for server requests using the XIO transport. The timeout is set in seconds. The valid range is timeout >= 1.

Parameters:
`timeout` - the timeout in seconds
Since:
8.6, XC10 2.5

---

getXIORequestTimeout

`int` **`getXIORequestTimeout`**`()`

Gets the current timeout that is allowed for requests to servers over the XIO transport to complete. The timeout is set in milliseconds.

Since:
8.6.0.2, XC10 2.5

---

setXIORequestTimeout

`void` **`setXIORequestTimeout`**`(int timeout)`

Sets the current timeout that is allowed for requests to servers over the XIO transport to complete. The timeout is set in milliseconds. The valid range is greater than 0.

Parameters:
`timeout` - the timeout in milliseconds
Since:
8.6.0.2, XC10 2.5

Package com.ibm.websphere.objectgrid.config

This package contains the interfaces and a factory class for creating ObjectGrid configuration objects programatically.

See:

Description

Interface Summary

BackingMapConfiguration	A BackingMapConfiguration object can be used to override BackingMap settings on the client side.
ConfigProperty	ConfigProperty can be used to attach properties to a `Plugin`.
ObjectGridConfiguration	An ObjectGridConfiguration object can be used to override ObjectGrid plugins on the client side.
Plugin	This interface represents an ObjectGrid or BackingMap plugin.
PluginType	Every `Plugin` has a PluginType.

Class Summary

ConfigPropertyType	ConfigPropertyType is used to set the type of an attribute on a Plugin.
ObjectGridConfigFactory	This is the configuration factory for ObjectGrid configuration entities.
QueryConfig	This QueryConfig represents a schema configuration for a QueryManager.
QueryMapping	A QueryMapping maps a Java class to a BackingMap and allows a map to be included in a query.
QueryRelationship	A QueryRelationship represents a relationship between two BackingMap value classes.

Exception Summary

ObjectGridConfigurationException	Thrown when a problem with the current configuration is found.

Package com.ibm.websphere.objectgrid.config Description

This package contains the interfaces and a factory class for creating ObjectGrid configuration objects programatically. The main use of this is by the objectgrid client to override serverside configuration.

Overview

ObjectGridManagerFactory has static methods to create the configuration objects. Using these configuration objects in conjuction with ObjectGridManager methods

- setOverrideObjectGridConfigurations(Map)
- putOverrideObjectGridConfigurations(String, List)

to override client configuration, before connecting to the objectgrid server.

com.ibm.websphere.objectgrid.config

Class QueryRelationship

java.lang.Object
  com.ibm.websphere.objectgrid.config.QueryRelationship

All Implemented Interfaces:

Serializable

```
public class QueryRelationship
extends Object
implements Serializable
```

A QueryRelationship represents a relationship between two BackingMap value classes. A BackingMap must have one class type defined in the value part of the map. A relationship can be established between two maps by mapping the source and target map's value classes.

The cardinality of the relationship is automatically determined by the type of the attribute field.

A relationship requires two classes, a relationship field, and optionally an inverse realtionship field.

For example: Two entities; Department and Employee, have the following bi-directional relationship:

1. One department has many employees. the collection field in the Department class is "emps"
2. An employee belongs to one department

```
public class Department {
    private int id;
    private Collection emps;

    public void setEmps(Collection emps) {
        this.emps = emps;
    }

    public Collection getEmps() {
        return emps;
    }
    ...
}

public class Employee {
    private int id;
    private Department dept;

    public void setDept(Department dept) {
        this.dept = dept;
    }

    public Department getDept() {
        return dept;
    }
```

```
}
```

Use the following method call to establish this bi-directional relationship.

```
queryConfig.addRelationship(new QueryRelationship(
    Department.class.getName(), Employee.class.getName(), "emps", "dept"));
```

Since:

WAS XD 6.1, XC10

See Also:

Serialized Form

Constructor Summary

QueryRelationship(String sourceClass, String targetClass, String relationshipField, String invRelationshipField) Constructor for creating a QueryRelationship instance.

Method Summary

boolean	**equals**(Object o)
String	**getInvRelationshipField**() Retrieve the inverse relationship attribute name of a bi-directional relationship.
String	**getRelationshipField**() Retrieve the name of the attribute in the source class that references the key of the target class.
String	**getSourceClass**() Retrieve the name of the class representing the source of a relationship.
String	**getTargetClass**() Retrieve the name of the class representing the target of a relationship.
int	**hashCode**()
void	**setInvRelationshipField**(String invRelationshipField) Set the inverse relationship attribute name of a bi-directional relationship.

void	**setRelationshipField**(String relationshipField) Set the name of the attribute in the source class that references the key of the target class.
void	**setSourceClass**(String sourceClass) Set the name of the class representing the source of a relationship.
void	**setTargetClass**(String targetClass) Set the name of the class representing the target of a relationship.
String	**toString**()

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

QueryRelationship

```
public QueryRelationship(String sourceClass,
                         String targetClass,
                         String relationshipField,
                         String invRelationshipField)
```

Constructor for creating a QueryRelationship instance.

The sourceClass, targetClass, and relationshipField must not be null.

Parameters:

`sourceClass` - the source class of the relationship
`targetClass` - the target class of the relationship
`relationshipField` - the attribute in the source class that references the key of the target class.
`invRelationshipField` - the attribute in the target class that references the key of the source class. This value is null if a bi-directional relationship does not exist.

Method Detail

getInvRelationshipField

```
public String getInvRelationshipField()
```

Retrieve the inverse relationship attribute name of a bi-directional relationship.

Returns:

the attribute name of the inverse side of a bi-directional relationship or null if the relationship is uni-directional.

---

setInvRelationshipField

public void **setInvRelationshipField**(String invRelationshipField)

Set the inverse relationship attribute name of a bi-directional relationship.

Parameters:

`invRelationshipField` - the attribute name of the inverse side of a bi-directional relationship or null if the relationship is uni-directional.

---

getRelationshipField

public String **getRelationshipField**()

Retrieve the name of the attribute in the source class that references the key of the target class.

Returns:

the name of the relationship attribute.

---

setRelationshipField

public void **setRelationshipField**(String relationshipField)

Set the name of the attribute in the source class that references the key of the target class.

Parameters:

`relationshipField` - the name of the relationship attribute.

---

getSourceClass

public String **getSourceClass**()

Retrieve the name of the class representing the source of a relationship.

Returns:

the source class

---

setSourceClass

public void **setSourceClass**(String sourceClass)

Set the name of the class representing the source of a relationship.

Parameters:

`sourceClass` - the source class

---

getTargetClass

public String **getTargetClass**()

Retrieve the name of the class representing the target of a relationship.

Returns:

the target class

---

setTargetClass

public void **setTargetClass**(String targetClass)

Set the name of the class representing the target of a relationship.

Parameters:

`targetClass` - the target class

---

equals

public boolean **equals**(Object o)

Overrides:

equals in class Object

See Also:

Object.equals(java.lang.Object)

---

hashCode

public int **hashCode**()

Overrides:

hashCode in class Object

See Also:

Object.hashCode()

---

toString

public String **toString**()

Overrides:

toString in class Object

---

com.ibm.websphere.objectgrid.config

Class QueryMapping

java.lang.Object
  com.ibm.websphere.objectgrid.config.QueryMapping

All Implemented Interfaces:
Serializable

```
public class QueryMapping
extends Object
implements Serializable
```

A QueryMapping maps a Java class to a BackingMap and allows a map to be included in a query. It also indicates whether the query engine should use a getter method or direct field access to access fields in the value class.

For example, class Department is the value class that is stored in the "DepartmentMap" BackingMap and the key is an Integer.

```
public class Department {
    private int id;
    private Collection emps;

    public void setEmps(Collection emps) {
        this.emps = emps;
    }

    public Collection getEmps() {
        return emps;
    }
    ...
}
```

The QueryMapping would be created as follows:

```
...
QueryConfig queryConfig = new QueryConfig();
queryConfig.addMapping(new QueryMapping(
    "DepartmentMap", Department.class.getName(), "id", QueryMapping.PROPERTY_ACCESS)
objectGrid.setQueryConfig(queryConfig);
...
```

Since:
WAS XD 6.1, XC10
See Also:
Serialized Form

Field Summary

s t a

static int	**FIELD_ACCESS** This constant indicates to use direct field access to read the field values
static int	**PROPERTY_ACCESS** This constant indicates to use JavaBean property-style get methods to read the field values from the Java object stored in the BackingMap.

Constructor Summary

QueryMapping() Default constructor.
QueryMapping(String mapName, String valueClass, String primaryKeyField) Constructor for creating a basic QueryMapping instance with a default access type of PROPERTY_ACCESS.
QueryMapping(String mapName, String valueClass, String primaryKeyField, int accessType) Constructor for creating a QueryMapping instance.

Method Summary

boolean	**equals**(Object o)
int	**getAccessType**() Retrieve the method in which the query engine will access the value class object stored in the BackingMap.
String	**getMapName**() Retrieve the BackingMap name associated with this mapping.
String	**getPrimaryKeyField**() Retrieve the name of the attribute in the value class object that is also the primary key of the BackingMap.
String	**getValueClass**() Retrieve the type of object that is stored in the BackingMap.
int	**hashCode**()

void	**setAccessType**(int accessType) Set the method in which the query engine will access the value class object stored in the BackingMap.
void	**setMapName**(String mapName) Set the BackingMap name associated with this mapping.
void	**setPrimaryKeyField**(String primaryKeyField) Set the name of the attribute in the value class object that is also the primary key of the BackingMap.
void	**setValueClass**(String valueClass) Set the type of object that is stored in the BackingMap.
String	**toString**()

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

FIELD_ACCESS

public static final int **FIELD_ACCESS**

This constant indicates to use direct field access to read the field values

See Also:
setAccessType(int), getAccessType(), Constant Field Values

PROPERTY_ACCESS

public static final int **PROPERTY_ACCESS**

This constant indicates to use JavaBean property-style get methods to read the field values from the Java object stored in the BackingMap. PROPERTY_ACCESS is the default.

See Also:
setAccessType(int), getAccessType(), Constant Field Values

Constructor Detail

QueryMapping

public **QueryMapping**()

Default constructor.

QueryMapping

```
public QueryMapping(String mapName,
                    String valueClass,
                    String primaryKeyField)
```

Constructor for creating a basic QueryMapping instance with a default access type of PROPERTY_ACCESS.

The mapName and valueClass must not be null.

Parameters:

`mapName` - the name of the BackingMap to map
`valueClass` - the class of object stored in the BackingMap's value.
`primaryKeyField` - the optional name of the primary key field of the class.

QueryMapping

```
public QueryMapping(String mapName,
                    String valueClass,
                    String primaryKeyField,
                    int accessType)
```

Constructor for creating a QueryMapping instance. The mapName and valueClass must not be null.

Parameters:

`mapName` - the name of the BackingMap to map
`valueClass` - the class of object stored in the BackingMap's value.
`primaryKeyField` - the optional name of the primary key field of the class.
`accessType` - the method (PROPERTY_ACCESS or FIELD_ACCESS) in which the query engine will access the persistent data in the value object.

Method Detail

getMapName

```
public String getMapName()
```

Retrieve the BackingMap name associated with this mapping.

Returns:

the BackingMap name.

---

setMapName

```
public void setMapName(String mapName)
```

Set the BackingMap name associated with this mapping.

---

getValueClass

```
public String getValueClass()
```

Retrieve the type of object that is stored in the BackingMap.

Returns:

the object type that is stored in the BackingMap's value.

---

setValueClass

`public void **setValueClass**(String valueClass)`

Set the type of object that is stored in the BackingMap.

---

getAccessType

`public int **getAccessType**()`

Retrieve the method in which the query engine will access the value class object stored in the BackingMap.

Returns:
Returns the accessType.
See Also:
PROPERTY_ACCESS, FIELD_ACCESS

---

setAccessType

`public void **setAccessType**(int accessType)`

Set the method in which the query engine will access the value class object stored in the BackingMap.

Parameters:
`accessType` - the accessType.
See Also:
PROPERTY_ACCESS, FIELD_ACCESS

---

getPrimaryKeyField

`public String **getPrimaryKeyField**()`

Retrieve the name of the attribute in the value class object that is also the primary key of the BackingMap.

This value is optional.

Returns:
the primaryKeyField.

---

setPrimaryKeyField

`public void **setPrimaryKeyField**(String primaryKeyField)`

Set the name of the attribute in the value class object that is also the primary key of the BackingMap.

Parameters:
`primaryKeyField` - the name of the primary key attribute or null if not set.

---

equals

`public boolean **equals**(Object o)`

Overrides:

equals in class Object

See Also:

Object.equals(java.lang.Object)

---

hashCode

public int **hashCode**()

Overrides:

hashCode in class Object

See Also:

Object.hashCode()

---

toString

public String **toString**()

Overrides:

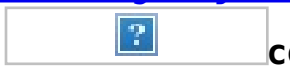
toString in class Object

---

com.ibm.websphere.objectgrid.config

Class QueryConfig

java.lang.Object



com.ibm.websphere.objectgrid.config.QueryConfig

All Implemented Interfaces:

Serializable

```
public class QueryConfig
extends Object
implements Serializable
```

This QueryConfig represents a schema configuration for a QueryManager. A QueryConfig object contains a set of QueryMapping objects and a set of QueryRelationship objects.

Users use addQuerySchema method to add a QueryMapping object and use addRelationship method to add a QueryRelationship object into this QueryConfig object.

Since:

WAS XD 6.1, XC10

See Also:

ObjectGrid.setQueryConfig(QueryConfig), QueryMapping, QueryRelationship, Serialized Form

---

Constructor Summary

QueryConfig()	Default constructor.

Method Summary

void	**addQueryMapping**(QueryMapping mapping) Add a QueryMapping to this QueryConfig
void	**addQueryRelationship**(QueryRelationship relation) Add a QueryRelationship to this QueryConfig
QueryMapping[]	**getQueryMappings**() Retrieve all added QueryMappings

QueryRelationship[]	**getQueryRelationships**() Retrieve all added QueryRelationships

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

QueryConfig

```
public QueryConfig()
```

Default constructor.

Method Detail

addQueryMapping

```
public void addQueryMapping(QueryMapping mapping)
```

Add a QueryMapping to this QueryConfig

Parameters:

`mapping` - the QueryMapping to add.

---

addQueryRelationship

```
public void addQueryRelationship(QueryRelationship relation)
```

Add a QueryRelationship to this QueryConfig

Parameters:

`relation` - the QueryRelationship to add.

---

getQueryMappings

```
public QueryMapping[] getQueryMappings()
```

Retrieve all added QueryMappings

Returns:

array of QueryMapping

---

getQueryRelationships

public QueryRelationship[] **getQueryRelationships**()

Retrieve all added QueryRelationships

Returns:

array of QueryRelationship

---

com.ibm.websphere.objectgrid.config

Interface PluginType

All Superinterfaces:

Serializable

```
public interface PluginType
extends Serializable
```

Every `Plugin` has a PluginType. The PluginType is specified during Plugin creation.

`ObjectGridConfiguration` objects support the following PluginTypes for overriding client-side ObjectGrid plugins:

- `PluginType.TRANSACTION_CALLBACK`
- `PluginType.OBJECTGRID_EVENT_LISTENER`
- `PluginType.COLLISION_ARBITER`

`BackingMapConfiguration` objects support the following PluginTypes for overriding client-side BackingMap plugins:

- `PluginType.EVICTOR`
- `PluginType.MAP_EVENT_LISTENER`

Since:

WAS XD 6.0.1.2, XC10

See Also:

`ObjectGridConfigFactory.createPlugin(PluginType, String)`

Field Summary

static PluginType	**COLLISION_ARBITER** `PluginType.COLLISION_ARBITER` can be used to attach a `CollisionArbiter` plugin to an ObjectGrid.
static PluginType	**EVICTOR** `PluginType.EVICTOR` can be used to attach an `Evictor` to a BackingMap.

[gInType](#)	
static [PluginType](#)	**MAP_EVENT_LISTENER** `PluginType.MAP_EVENT_LISTENER` can be used to attach a [MapEventListener](#) to a BackingMap.
static [PluginType](#)	**MAP_LIFECYCLE_LISTENER** `PluginType.MAP_LIFECYCLE_LISTENER` can be used to attach a `BackingMapLifecycleListener` plugin to an BackingMap
static [PluginType](#)	**MAP_SERIALIZER_PLUGIN** `PluginType.MAP_SERIALIZER_PLUGIN` can be used to attach a `MapSerializerPlugin` to a BackingMap to serialize keys.
static [PluginType](#)	**OBJECTGRID_EVENT_LISTENER** `PluginType.OBJECTGRID_EVENT_LISTENER` can be used to attach an [ObjectGridEventListener](#) plugin to an ObjectGrid
stat	

static PluginType	**OBJECTGRID_LIFECYCLE_LISTENER** PluginType.OBJECTGRID_LIFECYCLE_LISTENER can be used to attach an ObjectGridLifecycleListener plugin to an ObjectGrid
static PluginType	**TRANSACTION_CALLBACK** PluginType.TRANSACTION_CALLBACK can be used to attach a TransactionCallback plugin to an ObjectGrid.

Field Detail

OBJECTGRID_EVENT_LISTENER

static final PluginType **OBJECTGRID_EVENT_LISTENER**

PluginType.OBJECTGRID_EVENT_LISTENER can be used to attach an ObjectGridEventListener plugin to an ObjectGrid

---

TRANSACTION_CALLBACK

static final PluginType **TRANSACTION_CALLBACK**

PluginType.TRANSACTION_CALLBACK can be used to attach a TransactionCallback plugin to an ObjectGrid.

COLLISION_ARBITER

static final PluginType **COLLISION_ARBITER**

PluginType.COLLISION_ARBITER can be used to attach a CollisionArbiter plugin to an ObjectGrid.

OBJECTGRID_LIFECYCLE_LISTENER

static final PluginType **OBJECTGRID_LIFECYCLE_LISTENER**

PluginType.OBJECTGRID_LIFECYCLE_LISTENER can be used to attach an ObjectGridLifecycleListener plugin to an ObjectGrid

Since:
7.1.1

EVICTOR

static final PluginType **EVICTOR**

PluginType.EVICTOR can be used to attach an Evictor to a BackingMap.

MAP_EVENT_LISTENER

static final PluginType **MAP_EVENT_LISTENER**

PluginType.MAP_EVENT_LISTENER can be used to attach a MapEventListener to a BackingMap.

MAP_SERIALIZER_PLUGIN

static final PluginType **MAP_SERIALIZER_PLUGIN**

PluginType.MAP_SERIALIZER_PLUGIN can be used to attach a MapSerializerPlugin to a BackingMap to serialize keys.

Since:
7.1.1

MAP_LIFECYCLE_LISTENER

static final PluginType **MAP_LIFECYCLE_LISTENER**

PluginType.MAP_LIFECYCLE_LISTENER can be used to attach a BackingMapLifecycleListener plugin to an BackingMap

Since:
7.1.1

---

com.ibm.websphere.objectgrid.config

Interface Plugin

All Superinterfaces:

Serializable

```
public interface Plugin
extends Serializable
```

This interface represents an ObjectGrid or BackingMap plugin. An `ObjectGridConfiguration` object supports the following Plugins:

- `PluginType#OBJECTGRID_EVENT_LISTENER`
- `PluginType#TRANSACTION_CALLBACK`

A `BackingMapConfiguration` object supports the following Plugins:

- `PluginType#EVICTOR`
- `PluginType#MAP_EVENT_LISTENER`

A Plugin object has following attributes:

- pluginType: the support PluginTypes are listed above
- className: the plugin implementation class name. This class name will be used to construct the class object. A default constructor must be implemented.
- configProperties: the JavaBean properties for this plugin implementation object.

A plugin object can be created by using `ObjectGridConfigFactory.createPlugin(PluginType, String)` `method. Please refer to ObjectGridConfigFactory for detailed example.`

Since:

WAS XD 6.0.1.2, XC10

Method Summary

void	**addConfigProperty**(ConfigProperty configProp) Add a ConfigProperty to this object.
String	**getClassName**() Get the String representation of the class name of this Plugin
List	**getConfigProperties**() Get the ConfigProperty objects that have been set on this object.
Str	**getOSGiService**()

String	Get the OSGi service name configured for this Plugin.
PluginType	**getPluginType**() Get the PluginType for this Plugin.
void	**setClassName**(String className) The class name that is set must be an implementor of the PluginTypeImpl for this Plugin.
void	**setConfigProperties**(List configPropList) Set the ConfigProperties for this object
void	**setOSGiService**(String osgiService) Set the OSGi service name configured for this Plugin.
void	**setPluginType**(PluginType pluginType) Set the PluginType for this Plugin.

Method Detail

addConfigProperty

void **addConfigProperty**(ConfigProperty configProp)

Add a ConfigProperty to this object.

Parameters:
configProp - - ConfigProperty to add to this object

---

setConfigProperties

void **setConfigProperties**(List configPropList)

Set the ConfigProperties for this object

Parameters:
configPropList -

---

getConfigProperties

List **getConfigProperties**()

Get the ConfigProperty objects that have been set on this object.

Returns:
a List of the ConfigProperty objects that have been added to this object.
See Also:
ConfigProperty

---

getPluginType

PluginType **getPluginType**()

Get the PluginType for this Plugin.

Returns:
the PluginType for this Plugin

---

setPluginType

void **setPluginType**(PluginType pluginType)

Set the PluginType for this Plugin.

The ObjectGridConfiguration plugins include

- PluginType.TRANSACTION_CALLBACK
- PluginType.OBJECTGRID_EVENT_LISTENER

The BackingMapConfiguration plugins include

- PluginType.EVICTOR
- PluginType.MAP_EVENT_LISTENER

Parameters:
pluginType -

---

getClassName

String **getClassName**()

Get the String representation of the class name of this Plugin

Returns:
the String representation of the class name

---

setClassName

void **setClassName**(String className)

The class name that is set must be an implementor of the PluginTypeImpl for this Plugin. For example, if the type of this Plugin is PluginType.EVICTOR, then the className must be an implementor of the com.ibm.websphere.objectgrid.plugins.Evictor interface.

Parameters:
className - - the class name of the Class that implements the PluginType

---

getOSGiService

String **getOSGiService**()

Get the OSGi service name configured for this Plugin. If an OSGi service name is configured for this Plugin, the className configured for this Plugin is ignored.

Returns:
the OSGi service name configured for this Plugin.
Since:
7.1.1

---

setOSGiService

void **setOSGiService**(String osgiService)

Set the OSGi service name configured for this Plugin. If an OSGi service name is configured for this Plugin, the className configured for this Plugin is ignored.

Parameters:
osgiService - the OSGi service name configured for this Plugin.
Since:
7.1.1

com.ibm.websphere.objectgrid.config

Class ObjectGridConfigurationException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.config.ObjectGridConfigurationException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **ObjectGridConfigurationException**
extends ObjectGridException

Thrown when a problem with the current configuration is found. This exception may be thrown when the configuration specified in the deployment policy, ObjectGrid descriptor or security descriptor is incorrect.

Since:

7.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridConfigurationException() Constructs a new ObjectGridConfigurationException with null as its detail message.
ObjectGridConfigurationException(String message) Constructs a new ObjectGridConfigurationException with the specified detail message.
ObjectGridConfigurationException(String message, Throwable cause) Constructs a new ObjectGridConfigurationException with the specified detail message and cause.
ObjectGridConfigurationException(Throwable cause) Constructs a new ObjectGridConfigurationException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridConfigurationException

public **ObjectGridConfigurationException**()

Constructs a new ObjectGridConfigurationException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

ObjectGridConfigurationException

public **ObjectGridConfigurationException**(String message)

Constructs a new ObjectGridConfigurationException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

ObjectGridConfigurationException

public **ObjectGridConfigurationException**(Throwable cause)

Constructs a new ObjectGridConfigurationException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ObjectGridConfigurationException that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

ObjectGridConfigurationException

public **ObjectGridConfigurationException**(String message,
Throwable cause)

Constructs a new ObjectGridConfigurationException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in

this ObjectGridConfigurationException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`

---

com.ibm.websphere.objectgrid.config

Interface ObjectGridConfiguration

public interface **ObjectGridConfiguration**

An ObjectGridConfiguration object can be used to override ObjectGrid plugins on the client side. The `com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener` and the `com.ibm.websphere.objectgrid.plugins.TransactionCallback` Plugins can be overridden.

Since:

WAS XD 6.0.1.2, XC10

See Also:

ObjectGridEventListener, TransactionCallback

Method Summary

void	**addBackingMapConfiguration**(BackingMapConfiguration backingMapConfiguration) Add a BackingMapConfiguration to this ObjectGridConfiguration.
void	**addPlugin**(Plugin plugin) Add a `Plugin` to this ObjectGridConfiguration.
List	**getBackingMapConfigurations**() Get the List of BackingMapConfiguration objects that are attached to this ObjectGridConfiguration object
String	**getName**() Get the name of this ObjectGridConfiguration
List	**getPlugins**() Get the Plugins that have been attached to this ObjectGridConfiguration.
int	**getTxIsolation**() Retrieves the default transaction isolation level.
int	**getTxTimeout**() Gets the transaction timeout.
void	**setBackingMapConfigurations**(List backingMapConfigList) Set the BackingMapConfiguration objects for this ObjectGridConfiguration.

void	**setPlugins**(List pluginList) Set the Plugins for this ObjectGridConfiguration.
void	**setTxIsolation**(int level) Sets the default transaction isolation level for all sessions created by the ObjectGrid.
void	**setTxTimeout**(int seconds) Sets the transaction timeout

Method Detail

getName

String **getName**()

Get the name of this ObjectGridConfiguration

Returns:
the name of this ObjectGridConfiguration

---

addBackingMapConfiguration

void **addBackingMapConfiguration**(BackingMapConfiguration backingMapConfiguration)

Add a BackingMapConfiguration to this ObjectGridConfiguration.

Parameters:
`backingMapConfiguration` -

---

setBackingMapConfigurations

void **setBackingMapConfigurations**(List backingMapConfigList)

Set the BackingMapConfiguration objects for this ObjectGridConfiguration. Any BackingMapConfiguration objects that were previously attached to this ObjectGridConfiguration object will be overridden.

Parameters:
`backingMapConfigList` - - A List of `BackingMapConfiguration` objects.
See Also:
BackingMapConfiguration

---

getBackingMapConfigurations

List **getBackingMapConfigurations**()

Get the List of BackingMapConfiguration objects that are attached to this ObjectGridConfiguration object

Returns:
a List of `BackingMapConfiguration` objects
See Also:

BackingMapConfiguration

---

addPlugin

`void` **`addPlugin`**`(`Plugin `plugin)`

Add a `Plugin` to this ObjectGridConfiguration. The Plugins that can be overridden on a client-side ObjectGrid are `com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener` and `com.ibm.websphere.objectgrid.plugins.TransactionCallback`.

Parameters:

`plugin` -

See Also:

setPlugins(List), Plugin

---

setPlugins

`void` **`setPlugins`**`(`List `pluginList)`

Set the Plugins for this ObjectGridConfiguration. Any Plugins that were previously attached to this ObjectGridConfiguration object will be overridden.

Parameters:

`pluginList` - - a List of Plugins

See Also:

addPlugin(Plugin), Plugin

---

getPlugins

List **`getPlugins`**`()`

Get the Plugins that have been attached to this ObjectGridConfiguration.

Returns:

a List of `Plugin` objects

See Also:

Plugin

---

setTxTimeout

`void` **`setTxTimeout`**`(int seconds)`

Sets the transaction timeout

Parameters:

`seconds` -

Since:

7.1.0.3

See Also:

ObjectGrid.setTxTimeout(int)

---

getTxTimeout

`int` **`getTxTimeout`**`()`

Gets the transaction timeout. The value is in seconds.

Returns:
the transaction timeout in seconds
Since:
7.1.0.3
See Also:
ObjectGrid.getTxTimeout()

---

setTxIsolation

`void` **`setTxIsolation`**`(int level)`

Sets the default transaction isolation level for all sessions created by the ObjectGrid. The constants defined in the `Session` interface are the possible transaction isolation levels. The default is Session.TRANSACTION_REPEATABLE_READ.

Parameters:
`level` - one of the following `Session` constants: Session.TRANSACTION_READ_UNCOMMITTED, Session.TRANSACTION_READ_COMMITTED or Session.TRANSACTION_REPEATABLE_READ or 0 if the TransactionIsolation should not be set.
Since:
7.1.1

---

getTxIsolation

`int` **`getTxIsolation`**`()`

Retrieves the default transaction isolation level.

Returns:
the current transaction isolation level.
Since:
7.1.1
See Also:
setTxIsolation(int)

---

com.ibm.websphere.objectgrid.config

Class ObjectGridConfigFactory

java.lang.Object

com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory

---

public final class **ObjectGridConfigFactory**
extends Object

This is the configuration factory for ObjectGrid configuration entities. Users are expected to use static methods of this factory to create ObjectGrid configuration objects.

Here is a list of static methods used to create configuration objects:

- `createObjectGridConfiguration(String)`: create an ObjectGridConfiguration object
- `createBackingMapConfiguration(String)`: create a `BackingMapConfiguration` object by passing a backing map name
- `createConfigProperty(ConfigPropertyType, String, String)`: create a `ConfigProperty` object
- `createPlugin(PluginType, String)`: create a `Plugin` object

Below is an example of creating an ObjectGrid configuration. A Plugin is added to the `ObjectGridConfiguration` object: `com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener` plugin.

A `BackingMapConfiguration` called "myBackingMap" is then created and added to the `ObjectGridConfiguration`. This `BackingMapConfiguration` also has an Evictor Plugin configured.

Once the ObjectGridConfiguration object has been created , it can be used to call either of these methods

- `com.ibm.websphere.objectgrid.ObjectGridManager.putOverrideObjectGridConfigurations(String, List)`
- `com.ibm.websphere.objectgrid.ObjectGridManager.setOverrideObjectGridConfigurations(Map)`

to set configuration objects, prior to connecting.

```
// Create an ObjectGridConfiguration object
ObjectGridConfiguration ogConfig = ObjectGridConfigFactory.createObjectGridConfiguration(ogName);

// create ObjectGridEventListener plugin
 Plugin eventListener = ObjectGridConfigFactory.createPlugin(PluginType.OBJECTGRID_EVENT_LISTENER,
"com.ibm.test.MyOgEventListener");

 // Add plugin to ObjectGridConfiguration object
 ogConfig.addPlugin(eventListener);

 // Create a BackingMapConfiguration object
 BackingMapConfiguration bmConfig = ObjectGridConfigFactory.createBackingMapConfiguration("mybacki
ngMap");

 // Add BackingMapConfiguration object to ObjectGridConfiguration object
 ogConfig.addBackingMapConfiguration(bmConfig);

 // Set the number of buckets to 1000
 bmConfig.setNumberOfBuckets(1000);
```

```
// Create a Evictor plugin for this backing map.
Plugin evictor = ObjectGridConfigFactory.createPlugin(
BackingMapConfiguration.PLUGIN_EVICTOR,
com.acme.myEvictorImpl.class.getName());

// add Evictor Plugin to the BackingMapConfiguration
bmConfig.addPlugin(evictor);

// Create a ConfigProperty for the Evictor plugin
ConfigProperty sizeProperty = ObjectGridConfigFactory.createConfigProperty(
  ConfigPropertyType.INT_PRIM, "size", "153");

// Add the ConfigProperty to the Evictor plugin
evictor.setConfigProperty(sizeProperty);
```

Since:

WAS XD 6.0.1.2, XC10

See Also:

ObjectGridConfiguration, BackingMapConfiguration, Plugin, ConfigProperty

---

Constructor Summary

ObjectGridConfigFactory()

Method Summary

static BackingMapConfiguration	**createBackingMapConfiguration**(String backingMapConfigName) Create a BackingMapConfiguration object
static ConfigProperty	**createConfigProperty**(ConfigPropertyType configPropType, String name, String value) Create a ConfigPropety object for use on a Plugin.

Property	
static ObjectGridConfiguration	**createObjectGridConfiguration**(String objectGridConfigName) Create an ObjectGridConfiguration object.
static Plugin	**createOSGiServicePlugin**(PluginType pluginType, String osgiServiceName) Create an OSGi Service Plugin for a BackingMapConfiguration or an ObjectGridConfiguration
static Plugin	**createPlugin**(PluginType pluginType, String className) Create a Plugin for a BackingMapConfiguration or an ObjectGridConfiguration

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridConfigFactory

public **ObjectGridConfigFactory**()

Method Detail

createPlugin

```
public static Plugin createPlugin(PluginType pluginType,
                                   String className)
```

Create a Plugin for a BackingMapConfiguration or an ObjectGridConfiguration

Parameters:

`pluginType` - the PluginType.
`className` - of the Plugin implementation class to instantiate

Returns:

a Plugin instance

See Also:

PluginType.OBJECTGRID_EVENT_LISTENER, PluginType.TRANSACTION_CALLBACK, PluginType.COLLISION_ARBITER, PluginType.EVICTOR, PluginType.MAP_EVENT_LISTENER, PluginType.OBJECTGRID_LIFECYCLE_LISTENER, PluginType.MAP_LIFECYCLE_LISTENER

---

createObjectGridConfiguration

```
public static ObjectGridConfiguration createObjectGridConfiguration(String objectGridConfigName)
```

Create an ObjectGridConfiguration object.

Parameters:

`objectGridConfigName` - the name that will be assigned to this ObjectGridConfiguration object

Returns:

the ObjectGridConfiguration object

---

createBackingMapConfiguration

```
public static BackingMapConfiguration createBackingMapConfiguration(String backingMapConfigName)
```

Create a BackingMapConfiguration object

Parameters:

`backingMapConfigName` - the name to assign to this BackingMapConfiguration

Returns:

the BackingMapConfiguration object

---

createConfigProperty

```
public static ConfigProperty createConfigProperty(ConfigPropertyType configPropType,
                                                   String name,
                                                   String value)
```

Create a ConfigPropety object for use on a Plugin.

The Plugin should have a set method that corresponds to the name of this ConfigProperty. The method must accept a parameter of the ConfigPropertyType that is specified on this ConfigProperty. For example, if the name of this ConfigProperty is set to "size", and the type is `ConfigPropertyType.INT_PRIM`, then the Plugin must have the method setSize(int). The value of the ConfigProperty will be passed to the setter of the Plugin when an ObjectGrid is created based on this configuration.

Parameters:

`configPropType` - ConfigPropertyType of the ConfigProperty. Part of the set method's signature, the type of parameter the set method requires. Valid
`name` - of the ConfigProperty. It must correspond to the name of a set method on the Plugin.
`value` - of the ConfigProperty. This value will be passed to the set method on the Plugin.

Returns:

the ConfigProperty object

See Also:

ConfigPropertyType.STRING_JAVA_LANG, ConfigPropertyType.BOOLEAN_JAVA_LANG, ConfigPropertyType.BOOLEAN_PRIM, ConfigPropertyType.BYTE_JAVA_LANG, ConfigPropertyType.BYTE_PRIM, ConfigPropertyType.CHARACTER_JAVA_LANG, ConfigPropertyType.CHAR_PRIM, ConfigPropertyType.DOUBLE_JAVA_LANG, ConfigPropertyType.DOUBLE_PRIM, ConfigPropertyType.FLOAT_JAVA_LANG, ConfigPropertyType.FLOAT_PRIM, ConfigPropertyType.INTEGER_JAVA_LANG, ConfigPropertyType.INT_PRIM, ConfigPropertyType.LONG_JAVA_LANG, ConfigPropertyType.LONG_PRIM, ConfigPropertyType.SHORT_JAVA_LANG

---

createOSGiServicePlugin

```
public static Plugin createOSGiServicePlugin(PluginType pluginType,
                                              String osgiServiceName)
```

Create an OSGi Service Plugin for a BackingMapConfiguration or an ObjectGridConfiguration

Parameters:

`pluginType` - the PluginType.
`osgiServiceName` - the OSGi service name

Returns:

a Plugin instance

Since:

7.1.1

See Also:

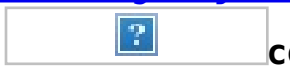
PluginType.OBJECTGRID_EVENT_LISTENER, PluginType.TRANSACTION_CALLBACK, PluginType.COLLISION_ARBITER, PluginType.EVICTOR, PluginType.MAP_EVENT_LISTENER, PluginType.OBJECTGRID_LIFECYCLE_LISTENER, PluginType.MAP_LIFECYCLE_LISTENER

---

com.ibm.websphere.objectgrid.config

Class ConfigPropertyType

java.lang.Object



com.ibm.websphere.objectgrid.config.ConfigPropertyType

All Implemented Interfaces:

Serializable

```
public final class ConfigPropertyType
extends Object
implements Serializable
```

ConfigPropertyType is used to set the type of an attribute on a Plugin. The Java primitives, their java.lang counterparts, and java.lang.String are the supported types.

Since:

WAS XD 6.0.1.2, XC10

See Also:

ObjectGridConfigFactory.createConfigProperty(ConfigPropertyType, String, String), Plugin.setPluginType(PluginType), Serialized Form

---

Field Summary

static ConfigPropertyType	**BOOLEAN_JAVA_LANG** ConfigPropertyType.BOOLEAN_JAVA_LANG can be used to set a property of type java.lang.Boolean on a plugin.
static Con	

static ConfigPropertyType	**BOOLEAN_PRIM** ConfigPropertyType.BOOLEAN_PRIM can be used to set a property of type `boolean` on a plugin.
static ConfigPropertyType	**BYTE_JAVA_LANG** ConfigPropertyType.BYTE_JAVA_LANG can be used to set a property of type `java.lang.Byte` on a plugin.
static ConfigPropertyType	**BYTE_PRIM** ConfigPropertyType.BYTE_PRIM can be used to set a property of type `byte` on a plugin.
static ConfigPropertyType	**CHAR_PRIM** ConfigPropertyType.CHAR_PRIM can be used to set a property of type `char` on a

[roperty Type](#)	plugin.
static [ConfigProperty Type](#)	**CHARACTER_JAVA_LANG** ConfigPropertyType.CHARACTER_JAVA_LANG can be used to set a property of type `java.lang.Character` on a plugin. **DOUBLE_JAVA_LANG** ConfigPropertyType.DOUBLE_JAVA_LANG can be used to set a property of type `java.lang.Double` on a plugin.
static [ConfigProp erty Type](#)	**DOUBLE_PRIM** ConfigPropertyType.DOUBLE_PRIM can be used to set a property of type `double` on a plugin.

[ertyType](#)	
staticConfigPropertyType	**FLOAT_JAVA_LANG** ConfigPropertyType.FLOAT_JAVA_LANG can be used to set a property of type `java.lang.Float` on a plugin.
staticConfigPropertyType	**FLOAT_PRIM** ConfigPropertyType.FLOAT_PRIM can be used to set a property of type `float` on a plugin.
staticConfigPropertyType	**INT_PRIM** ConfigPropertyType.INT_PRIM can be used to set a property of type `int` on a plugin.

[Type](#)	
static [ConfigPropertyType](#)	**INTEGER_JAVA_LANG** ConfigPropertyType.INTEGER_JAVA_LANG can be used to set a property of type `java.lang.Integer` on a plugin.
static [ConfigPropertyType](#)	**LONG_JAVA_LANG** ConfigPropertyType.LONG_JAVA_LANG can be used to set a property of type `java.lang.Long` on a plugin.
static [ConfigPropertyType](#)	**LONG_PRIM** ConfigPropertyType.LONG_PRIM can be used to set a property of type long on a plugin.

static ConfigPropertyType	**SHORT_JAVA_LANG** ConfigPropertyType.SHORT_JAVA_LANG can be used to set a property of type `java.lang.Short` on a plugin.
static ConfigPropertyType	**SHORT_PRIM** ConfigPropertyType.SHORT_PRIM can be used to set a property of type `short` on a plugin.
static ConfigPropertyType	**STRING_JAVA_LANG** ConfigPropertyType.STRING_JAVA_LANG can be used to set a property of type `java.lang.String` on a plugin.

Method Summary

boolean	**equals**(Object o)
int	**getId**() Get the raw value of this ConfigPropertyType.
int	**hashCode**()
String	**toString**()
static ConfigPropertyType	**valueOf**(byte id) Given the raw value of a ConfigPropertyType, this method returns a ConfigPropertyType object, or null if the raw value does not match an existing type.

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

INTEGER_JAVA_LANG

public static final ConfigPropertyType **INTEGER_JAVA_LANG**

ConfigPropertyType.INTEGER_JAVA_LANG can be used to set a property of type `java.lang.Integer` on a plugin.

INT_PRIM

public static final ConfigPropertyType **INT_PRIM**

ConfigPropertyType.INT_PRIM can be used to set a property of type `int` on a plugin.

BOOLEAN_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`BOOLEAN_JAVA_LANG`**

> ConfigPropertyType.BOOLEAN_JAVA_LANG can be used to set a property of type `java.lang.Boolean` on a plugin.

BOOLEAN_PRIM

`public static final` [ConfigPropertyType](#) **`BOOLEAN_PRIM`**

> ConfigPropertyType.BOOLEAN_PRIM can be used to set a property of type `boolean` on a plugin.

CHARACTER_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`CHARACTER_JAVA_LANG`**

> ConfigPropertyType.CHARACTER_JAVA_LANG can be used to set a property of type `java.lang.Character` on a plugin.

CHAR_PRIM

`public static final` [ConfigPropertyType](#) **`CHAR_PRIM`**

> ConfigPropertyType.CHAR_PRIM can be used to set a property of type `char` on a plugin.

BYTE_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`BYTE_JAVA_LANG`**

> ConfigPropertyType.BYTE_JAVA_LANG can be used to set a property of type `java.lang.Byte` on a plugin.

BYTE_PRIM

`public static final` [ConfigPropertyType](#) **`BYTE_PRIM`**

> ConfigPropertyType.BYTE_PRIM can be used to set a property of type `byte` on a plugin.

SHORT_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`SHORT_JAVA_LANG`**

> ConfigPropertyType.SHORT_JAVA_LANG can be used to set a property of type `java.lang.Short` on a plugin.

SHORT_PRIM

`public static final` [ConfigPropertyType](#) **`SHORT_PRIM`**

> ConfigPropertyType.SHORT_PRIM can be used to set a property of type `short` on a plugin.

LONG_JAVA_LANG

`public static final ConfigPropertyType` **`LONG_JAVA_LANG`**

ConfigPropertyType.LONG_JAVA_LANG can be used to set a property of type `java.lang.Long` on a plugin.

LONG_PRIM

`public static final ConfigPropertyType` **`LONG_PRIM`**

ConfigPropertyType.LONG_PRIM can be used to set a property of type long on a plugin.

FLOAT_JAVA_LANG

`public static final ConfigPropertyType` **`FLOAT_JAVA_LANG`**

ConfigPropertyType.FLOAT_JAVA_LANG can be used to set a property of type `java.lang.Float` on a plugin.

FLOAT_PRIM

`public static final ConfigPropertyType` **`FLOAT_PRIM`**

ConfigPropertyType.FLOAT_PRIM can be used to set a property of type `float` on a plugin.

DOUBLE_JAVA_LANG

`public static final ConfigPropertyType` **`DOUBLE_JAVA_LANG`**

ConfigPropertyType.DOUBLE_JAVA_LANG can be used to set a property of type `java.lang.Double` on a plugin.

DOUBLE_PRIM

`public static final ConfigPropertyType` **`DOUBLE_PRIM`**

ConfigPropertyType.DOUBLE_PRIM can be used to set a property of type `double` on a plugin.

STRING_JAVA_LANG

`public static final ConfigPropertyType` **`STRING_JAVA_LANG`**

ConfigPropertyType.STRING_JAVA_LANG can be used to set a property of type `java.lang.String` on a plugin.

Method Detail

valueOf

`public static final ConfigPropertyType` **`valueOf`**`(byte id)`

Given the raw value of a ConfigPropertyType, this method returns a ConfigPropertyType object, or null if the raw value does not match an existing type. This method is used to

deserialize this object.

Parameters:
`id` - the raw value of a ConfigPropertyType
Returns:
the ConfigPropertyType corresponding to the raw input value
Since:
8.6, XC10 2.5

---

equals

`public boolean **equals**(Object o)`

Overrides:
`equals` in class `Object`

---

hashCode

`public int **hashCode**()`

Overrides:
`hashCode` in class `Object`

---

getId

`public int **getId**()`

Get the raw value of this ConfigPropertyType. This method is used to serialize this object.

Returns:
the raw value of this ConfigPropertyType.

---

toString

`public String **toString**()`

Overrides:
`toString` in class `Object`

---

com.ibm.websphere.objectgrid.config

Interface ConfigProperty

All Superinterfaces:

Serializable

```
public interface ConfigProperty
extends Serializable
```

ConfigProperty can be used to attach properties to a `Plugin`. A ConfigProperty has the following attributes:

- name: the property name
- value: the property value
- configPropertyType: the configuration property type

This ConfigProperty can be used to set the properties of a plugin. The name of the property should follow JaveBean convention. That is, for every property, there should be a corresponding set method in the plugin class.

Users can use `com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory.createConfigProperty(ConfigPropertyType, String, String)` to create a ConfigProperty object.

```
ConfigProperty evictorNameProp =
ObjectGridConfigFactory.createConfigProperty(ConfigProperty.STRING_JAVA_LANG, "evictorName",
"evictor1");
```
This creates a property "evictorName" with value "evictor1", and the type is `java.lang.String`. Use the `com.ibm.websphere.objectgrid.config.Plugin#addConfigProperty(ConfigProperty)` method to attach a ConfigProperty to a Plugin. When the Plugin is created, each ConfigProperty will have its corresponding set method called.

Continuing with the example above, attach the ConfigProperty to an Evictor Plugin.
`evictorPlugin.addConfigProperty(evictorNameProp);`

When this Evictor Plugin is created, the `setEvictorName(String) method will be called with the value "evictor1".`

Since:
WAS XD 6.0.1.2, XC10

Method Summary

ConfigPropertyType	**getConfigPropertyType**() Get the ConfigPropertyType of this object.

String	**getName**() Get the name of this object.
String	**getValue**() Get the value that is assigned to this ConfigProperty.
void	**setConfigPropertyType**(ConfigPropertyType configPropType) Set the ConfigPropertyType for this object.
void	**setName**(String name) Set the name of this object.
void	**setValue**(String value) Set the value of this ConfigProperty.

Method Detail

setConfigPropertyType

void **setConfigPropertyType**(ConfigPropertyType configPropType)

Set the ConfigPropertyType for this object. The Java primitives, their java.lang counterparts, and java.lang.String are the supported ConfigPropertyTypes.

Parameters:
configPropType -

See Also:
ConfigPropertyType.INTEGER_JAVA_LANG, ConfigPropertyType.INT_PRIM, ConfigPropertyType.BOOLEAN_JAVA_LANG, ConfigPropertyType.BOOLEAN_PRIM, ConfigPropertyType.CHARACTER_JAVA_LANG, ConfigPropertyType.CHAR_PRIM, ConfigPropertyType.BYTE_JAVA_LANG, ConfigPropertyType.BYTE_PRIM, ConfigPropertyType.SHORT_JAVA_LANG, ConfigPropertyType.SHORT_PRIM, ConfigPropertyType.LONG_JAVA_LANG, ConfigPropertyType.LONG_PRIM, ConfigPropertyType.FLOAT_JAVA_LANG, ConfigPropertyType.FLOAT_PRIM, ConfigPropertyType.DOUBLE_JAVA_LANG, ConfigPropertyType.DOUBLE_PRIM, ConfigPropertyType.STRING_JAVA_LANG

---

getConfigPropertyType

ConfigPropertyType **getConfigPropertyType**()

Get the ConfigPropertyType of this object.

Returns:
the ConfigPropertyType for this object

---

setValue

void **setValue**(String value)

Set the value of this ConfigProperty. This String value will be converted to the proper type, based on ConfigPropertyType assigned to this ConfigProperty

Parameters:
value - - will be converted to type and passed to the setter on the plugin

---

getValue

String **getValue**()

Get the value that is assigned to this ConfigProperty. This is the value that will be passed to the set method on the plugin.

Returns:
Returns the value.

---

setName

void **setName**(String name)

Set the name of this object. The Plugin that this ConfigProperty is attached to should have a setter that corresponds to this name. For example, if "size" is passed in as the name, then the Plugin must have a "setSize" method.

Parameters:
name - - name of the property

---

getName

String **getName**()

Get the name of this object. The name must have a corresponding set method on the Plugin.

Returns:
Returns the name.

---

com.ibm.websphere.objectgrid.config

Interface BackingMapConfiguration

public interface **BackingMapConfiguration**

A BackingMapConfiguration object can be used to override BackingMap settings on the client side. The `com.ibm.websphere.objectgrid.plugins.Evictor` and the `com.ibm.websphere.objectgrid.plugins.MapEventListener` Plugins can be overridden. Other Evictor related settings can be adjusted as well as client, near-cache specific options.

Use the `com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory.createBackingMapConfiguration(String)` method to create a BackingMapConfiguration

Since:

WAS XD 6.0.1.2, XC10

See Also:

Evictor, MapEventListener, Plugin, ObjectGridConfigFactory

Method Summary

void	**addPlugin**(Plugin plugin) Add a `Plugin` to this BackingMapConfiguration.
String	**getEvictionTriggers**() Gets the list of eviction triggers for this BackingMapConfiguration.
com.ibm.websphere.objectgrid.plugins.KeyOutputFormat	**getKeyOutputFormat**() Retrieves the data format for all data access APIs that return cache keys.

com.ibm.websphere.objectgrid.OutputFormat	
String	**getName**() Get the name of this BackingMapConfiguration
int	**getNumberOfBuckets**() **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the [isNearCacheEnabled()](#) flag to disable the near cache in the ObjectGrid configuration XML file.*
List	**getPlugins**() Get the Plugins that have been attached to this BackingMapConfiguration.
int	**getTimeToLive**() Gets the "time to live" for each map entry.
TTLType	**getTtlEvictorType**() Gets the "time to live" Evictor type for this BackingMapConfiguration.
com.ibm.websphere.objectgrid.OutputFormat	**getValueOutputFormat**() Retrieves the data format for all data access APIs that return cache values.

.OutputFormat	
Boolean	**isNearCacheEnabled**() If true, the client local cache is enabled for supported configurations.
Boolean	**isNearCacheInvalidationEnabled**() If true, clients with local caches are automatically invalidated when the data grid map is updated.
Boolean	**isNearCacheLastAccessTTLSyncEnabled**() If true, clients automatically send time-to-live access information to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.
void	**setEvictionTriggers**(String evictionTriggers) Sets the eviction triggers for this BackingMapConfiguration.
void	**setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat) Sets the data format for all data access APIs that return cache keys.
void	**setNearCacheEnabled**(Boolean nearCacheEnabled) Enables or disables the client local cache for supported configurations.
void	**setNearCacheInvalidationEnabled**(Boolean nearCacheInvalidationEnabled) Set to true to enable client near cache invalidation.
void	**setNearCacheLastAccessTTLSyncEnabled**(Boolean nearCacheLastAccessTTLSyncEnabled) Enables or disables time-to-live access information synchronization to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.
void	**setNumberOfBuckets**(int numBuckets) **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the isNearCacheEnabled() flag to disable the near cache in the ObjectGrid configuration XML file.*
void	**setPlugins**(List pluginList) Set the Plugins for this BackingMapConfiguration.

void	**setTimeToLive**(int seconds) Sets "time to live" of each BackingMap entry in seconds.
void	**setTtlEvictorType**(TTLType ttlEvictorType) Set the "time to live" Evictor type for this BackingMapConfiguration.
void	**setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat) Sets the data format for all data access APIs that return cache values.

Method Detail

getName

String **getName**()

Get the name of this BackingMapConfiguration

Returns:
The name of this BackingMapConfiguration

---

addPlugin

void **addPlugin**(Plugin plugin)

Add a `Plugin` to this BackingMapConfiguration. The Plugins that can be overridden on a client-side BackingMap are `com.ibm.websphere.objectgrid.plugins.Evictor` and `com.ibm.websphere.objectgrid.plugins.MapEventListener`.

Parameters:
`plugin` -
See Also:
setPlugins(List)

---

setPlugins

void **setPlugins**(List pluginList)

Set the Plugins for this BackingMapConfiguration. Any Plugins that were previously attached to this BackingMapConfiguration object will be overridden.

Parameters:
`pluginList` - - a List of Plugins
See Also:
addPlugin(Plugin)

---

getPlugins

List **getPlugins**()

Get the Plugins that have been attached to this BackingMapConfiguration.

Returns:

a List of `Plugin` objects

---

getNumberOfBuckets

`int` **`getNumberOfBuckets`**`()`

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*

Gets the number of buckets defined for this BackingMapConfiguration.

Returns:

the number of buckets defined

---

setNumberOfBuckets

`void` **`setNumberOfBuckets`**`(int numBuckets)`

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*

Sets the number of buckets for this BackingMapConfiguration. This will be used by the `BackingMap`.

The `BackingMap` implementation uses a hash map for its implementation. If there are a lot of entries in the `BackingMap` then more buckets means better performance because the risk of collisions is lower as the number of buckets grows. More buckets also means more concurrency.

Parameters:

`numBuckets` -

See Also:

`BackingMap.setNumberOfBuckets(int)`

---

getTimeToLive

`int` **`getTimeToLive`**`()`

Gets the "time to live" for each map entry. The value is in seconds.

Returns:

the "time to live" in seconds

---

setTimeToLive

`void` **`setTimeToLive`**`(int seconds)`

Sets "time to live" of each BackingMap entry in seconds.

If this method is not called, the lifetime of an entry is forever (or until the application explicitly removes or invalidates the entry, or a user defined `Evictor` evicts the entry).

Parameters:

`seconds` -

---

getTtlEvictorType

TTLType **getTtlEvictorType**()

Gets the "time to live" Evictor type for this BackingMapConfiguration. If setTtlEvictorType was not called, this method will return null and the BackingMap based off this BackingMapConfiguration will use `TTLType.NONE`

Returns:
the "time to live" Evictor type or null if `setTtlEvictorType(TTLType) was not called`
See Also:
setTimeToLive(int)

---

setTtlEvictorType

void **setTtlEvictorType**(TTLType ttlEvictorType)

Set the "time to live" Evictor type for this BackingMapConfiguration. This is used to determine how expiration time of a BackingMap entry is computed.

If this method is not called, TTLType.NONE is used to indicate the map entry has no expiration time (e.g. is allowed to live until explicitly removed or invalidated by the application, or evicted by a user defined Evictor).

Parameters:
ttlEvictorType -
See Also:
BackingMap.setTtlEvictorType(TTLType)

---

getEvictionTriggers

String **getEvictionTriggers**()

Gets the list of eviction triggers for this BackingMapConfiguration.

See BackingMap for a list of valid eviction triggers.

Returns:
a semicolon separated list of eviction triggers or null if setEvictionTriggers(String) was not called
Since:
WAS XD 6.1.0.3

---

setEvictionTriggers

void **setEvictionTriggers**(String evictionTriggers)

Sets the eviction triggers for this BackingMapConfiguration. All evictors will use the eviction supplied triggers.

See BackingMap for a list of valid eviction triggers.

Parameters:
evictionTriggers - a semicolon separated list of eviction triggers
Since:
WAS XD 6.1.0.3

---

isNearCacheInvalidationEnabled

Boolean **isNearCacheInvalidationEnabled**()

If true, clients with local caches are automatically invalidated when the data grid map is updated.

Returns:
true if client near cache invalidation is enabled, false if client near cache invalidation is disabled, or null if the override is not specified.
Since:
8.6, XC10 2.5

---

setNearCacheInvalidationEnabled

void **setNearCacheInvalidationEnabled**(Boolean nearCacheInvalidationEnabled)

Set to true to enable client near cache invalidation. When enabled, the client will receive events from the remote data grid to invalidate data from the local cache.

Parameters:
nearCacheInvalidationEnabled - If true, the client near cache invalidation is enabled. If false, invalidation is disabled. If null, the override is not specified and the client will use the setting from the remote data grid.
Since:
8.6, XC10 2.5

---

isNearCacheLastAccessTTLSyncEnabled

Boolean **isNearCacheLastAccessTTLSyncEnabled**()

If true, clients automatically send time-to-live access information to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.

Returns:
True if last-access time-to-live information is sent to the remote data grid, false if last-access time information is not sent, or null if the override is not specified.
Since:
8.6, XC10 2.5

---

setNearCacheLastAccessTTLSyncEnabled

void **setNearCacheLastAccessTTLSyncEnabled**(Boolean nearCacheLastAccessTTLSyncEnabled)

Enables or disables time-to-live access information synchronization to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.

Parameters:
nearCacheLastAccessTTLSyncEnabled - If true, the last-access time-to-live information is sent to the remote data grid. If false, the last-access information is not sent. If null, the override is not specified and the client will use the setting from the remote data grid.
Since:
8.6, XC10 2.5

---

isNearCacheEnabled

Boolean **isNearCacheEnabled**()

If true, the client local cache is enabled for supported configurations. The client near cache can only be enabled when using optimistic locking or when locking is disabled.

Returns:
True if the client near cache is enabled, false if the near cache is disabled, or null of the override is not specified.
Since:
8.6, XC10 2.5

---

setNearCacheEnabled

void **setNearCacheEnabled**(Boolean nearCacheEnabled)

Enables or disables the client local cache for supported configurations. The client near cache can only be enabled when using optimistic locking or when locking is disabled.

Parameters:
nearCacheEnabled - If true, the client local cache is enabled for supported configurations. If false, the client local cache is disabled. If null, the override is not specified and the client will use the setting from the remote data grid.

---

getKeyOutputFormat

com.ibm.websphere.objectgrid.OutputFormat **getKeyOutputFormat**()

Retrieves the data format for all data access APIs that return cache keys.

Returns:
the data format or null if the default should be used.
Since:
8.6, XC10 2.5

---

setKeyOutputFormat

void **setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat)

Sets the data format for all data access APIs that return cache keys.

Parameters:
dataFormat - the data format to use or null to use the default.
Since:
8.6, XC10 2.5

---

getValueOutputFormat

com.ibm.websphere.objectgrid.OutputFormat **getValueOutputFormat**()

Retrieves the data format for all data access APIs that return cache values.

Returns:
the data format.
Since:
8.6, XC10 2.5

---

setValueOutputFormat

void **setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat)

Sets the data format for all data access APIs that return cache values.

Parameters:
dataFormat - the data format to use or null to use the default.
Since:
8.6, XC10 2.5

Package com.ibm.websphere.objectgrid.continuousquery

Interface Summary

ContinuousQueryCache<KeyType, ValueType>	A Continuous Query Cache contains the keys and optionally the values that match a defined continuous query.
ContinuousQueryFilter<KeyType,ValueType,AttributeType,MatchType>	An interface which provides an abstraction to check if an object matches the criteria defined in a filter.
ContinuousQueryListener<KeyType,ValueType>	This interface should be implemented to get a callback for a continuous query.
ContinuousQueryManager	The management interface used to define new continuous queries, retrieve an existing query, or destroy a query.
ContinuousQueryNotificationEvent<KeyType,ValueType>	An interface which defines notification metadata that is sent to continuous query listeners when the result set of a defined continous query changes.
ContinuousQueryTopic<KeyType,ValueType>	A Continuous Query Topic embodies the client side view of a defined Continuous Query.

Class Summary

ContinuousQueryManagerFactory	A factory class for obtaining the ContinuousQueryManager.

Enum Summary

ContinuousQueryNotificationEvent.Reason	Indicates the reason a notification event has occurred for this cache entry.

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryTopic<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried
`ValueType` - Type of the value object for the map being queried

`public interface` **`ContinuousQueryTopic<KeyType,ValueType>`**

A Continuous Query Topic embodies the client side view of a defined Continuous Query. The topic gives access to the client-side cached results of the continuous query and allows for management of Continuous Query Listeners for this topic.

Since:
8.6, XC10 2.5

Method Summary

void	**addListener**(ContinuousQueryListener<KeyType,ValueType> listener) Add a Continuous Query Listener to this topic.
Collection<ContinuousQueryListener<KeyType,ValueType>>	**getAllListeners**() Returns a shallow copy of the current set of listeners registered for this query topic.

[ContinuousQueryCache](#)<[KeyType](#), [ValueType](#)>	**[getCache](#)**() Return a continuous query cache object with methods to access the keys and values stored in the cache.
[String](#)	**[getName](#)**() Returns the generated unique name for this continuous query.
com.ibm.websphere	

e.objectgrid.OutputFormat	**getOutputFormat**() Return the format this query will use for keys and values.
List<Integer>	**getPartitions**() Return the list of partitions this query is defined against.
boolean	**isKeysOnlyCache**() Return true if the cache for this continuous query contains only keys.
boolean	**noCache**() Return true if the no caching option is enabled for this Continuous Query.
void	**removeAllListeners**() Removes all registered listeners from this query topic.
boolean	**removeListener**(ContinuousQueryListener<KeyType,ValueType> listener) Remove the listener from this continuous query's the set of listeners.

Method Detail

addListener

void **addListener**(ContinuousQueryListener<KeyType,ValueType> listener)

Add a Continuous Query Listener to this topic. The listener will be called when this topic receives updates to the query. There is no guarantee of the order that listeners will be called. Multiple adds of the same listener will result in only one call to the listener.

Parameters:

`listener` - the listener to add to this continuous query's set of listeners.

---

removeListener

boolean **removeListener**(ContinuousQueryListener<KeyType,ValueType> listener)

Remove the listener from this continuous query's the set of listeners. In flight notifications to this listener may continue after this method returns.

Parameters:

`listener` - - the continuous query listener to remove.

Returns:

true if the listener was removed.

---

getCache

ContinuousQueryCache<KeyType,ValueType> **getCache**()

Return a continuous query cache object with methods to access the keys and values stored in the cache.

Returns:

a Continuous Query Cache object for this topic.

---

getName

String **getName**()

Returns the generated unique name for this continuous query.

Returns:

a unique string.

---

isKeysOnlyCache

boolean **isKeysOnlyCache**()

Return true if the cache for this continuous query contains only keys. Return false if it also contains the value associated with the key.

Returns:

true if the cache does not contain values.

---

noCache

boolean **noCache**()

Return true if the no caching option is enabled for this Continuous Query.

Returns:

true if no keys or values are being cached.

getOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getOutputFormat`**`()`

Return the format this query will use for keys and values.

Returns:
the OutputFormat specified when this query was defined.

See Also:
`OutputFormat`

---

getAllListeners

`Collection<ContinuousQueryListener<KeyType,ValueType>>` **`getAllListeners`**`()`

Returns a shallow copy of the current set of listeners registered for this query topic.

Returns:
a new collection object containing all currently registered listeners.

---

removeAllListeners

`void` **`removeAllListeners`**`()`

Removes all registered listeners from this query topic. In flight notifications to listeners may continue after this method returns.

---

getPartitions

`List<Integer>` **`getPartitions`**`()`

Return the list of partitions this query is defined against. If the return value is null or an empty list, this query is defined against all partitions of the map.

Returns:
the array of partitions

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryNotificationEvent<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object in this event, from the key type used in the map being queried

`ValueType` - Type of the value object in this event, from the value type used in the map being queried

---

`public interface` **`ContinuousQueryNotificationEvent<KeyType,ValueType>`**

An interface which defines notification metadata that is sent to continuous query listeners when the result set of a defined continous query changes.

Since:

8.6, XC10 2.5

Nested Class Summary

static class	**ContinuousQueryNotificationEvent.Reason** Indicates the reason a notification event has occurred for this cache entry.

Method Summary

KeyType	**getKey**() Retrieve the key of the cache entry that triggered this notification.
ContinuousQu	

[EntryNotificationEvent.Reason](#)	**getReason**() Retrieve the reason for this notification event
com.ibm.websphere.objectgrid.SessionHandle	**getSessionHandle**() Retrieve a SessionHandle object which can be used to retrieve the cache entry that triggered this notification.
[ValueType](#)	**getValue**() Retrieve the value of the cache entry that triggered this notification, if the query was

[V] [p] [e]	configured to return the value object for matches.
boolean	**[isKeysOnly]**() If true, this notification only includes the key of the matching cache entry.

Method Detail

getKey

[KeyType] **getKey**()

Retrieve the key of the cache entry that triggered this notification. This object is a reference to the object stored in the [ContinuousQueryCache], if the cache has been enabled. Therefore it should not be modified.

Returns:
the cache entry key

---

getValue

[ValueType] **getValue**()

Retrieve the value of the cache entry that triggered this notification, if the query was configured to return the value object for matches. This object is a reference to the object stored in the [ContinuousQueryCache], if the cache has been enabled. Therefore it should not be modified.

Returns:
the cache entry value

---

getSessionHandle

com.ibm.websphere.objectgrid.SessionHandle **getSessionHandle**()

Retrieve a SessionHandle object which can be used to retrieve the cache entry that triggered this notification. This method can only be invoked when using a PER_CONTAINER placement strategy for the map.

Returns:
the SessionHandle for the partition that owns the cache entry.
See Also:
MapSet.getPlacementStrategy(), [Session.setSessionHandle(SessionHandle)]

---

getReason

[ContinuousQueryNotificationEvent.Reason] **getReason**()

Retrieve the reason for this notification event

Returns:

the Reason enum

---

isKeysOnly

`boolean **isKeysOnly**()`

If true, this notification only includes the key of the matching cache entry. If false, this notification also includes the value

Returns:
whether or not this notification includes the cache value.
See Also:
getValue()

---

com.ibm.websphere.objectgrid.continuousquery

Enum ContinuousQueryNotificationEvent.Reason

java.lang.Object

java.lang.Enum<ContinuousQueryNotificationEvent.Reason>

com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryNotificationEvent.Reason

All Implemented Interfaces:

Serializable, Comparable<ContinuousQueryNotificationEvent.Reason>

Enclosing interface:

ContinuousQueryNotificationEvent<KeyType,ValueType>

```
public static enum ContinuousQueryNotificationEvent.Reason
extends Enum<ContinuousQueryNotificationEvent.Reason>
```

Indicates the reason a notification event has occurred for this cache entry.

Enum Constant Summary

ADDED The cache entry was inserted into the grid and matches the query, or an entry which did not match the query was updated such that it now matches the query.
CLEAR A cache entry which matched the query was removed from the grid due to a map clear operation.
REMOVED The cache entry, which previously matched the query, was removed from the grid or updated such that it no longer matches.
UPDATED The cache entry, which previously matched the query, was updated and still matches the query.

Method Summary

static Continuous	

QueryNotificationEvent.Reason	**valueOf**(String name) Returns the enum constant of this type with the specified name.
static QueryNotificationEvent.Reason[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

ADDED

```
public static final ContinuousQueryNotificationEvent.Reason ADDED
```

The cache entry was inserted into the grid and matches the query, or an entry which did not match the query was updated such that it now matches the query.

REMOVED

```
public static final ContinuousQueryNotificationEvent.Reason REMOVED
```

The cache entry, which previously matched the query, was removed from the grid or updated such that it no longer matches.

UPDATED

```
public static final ContinuousQueryNotificationEvent.Reason UPDATED
```

The cache entry, which previously matched the query, was updated and still matches the query. Note: The result of getValue() for REMOVE events will always be null, even if the query was defined to include values.

See Also:

ContinuousQueryManager.defineContinuousQuery(java.lang.String, com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryFilter, boolean, boolean, boolean, java.util.Collection>, boolean, com.ibm.websphere.objectgrid.OutputFormat, java.util.List)

CLEAR

```
public static final ContinuousQueryNotificationEvent.Reason CLEAR
```

A cache entry which matched the query was removed from the grid due to a map clear operation.

Method Detail

values

```
public static ContinuousQueryNotificationEvent.Reason[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (ContinuousQueryNotificationEvent.Reason c : ContinuousQueryNotificationEvent.Reason.valu
es())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

---

valueOf

`public static ContinuousQueryNotificationEvent.Reason **valueOf**(String name)`

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`IllegalArgumentException` - if this enum type has no constant with the specified name

`NullPointerException` - if the argument is null

---

com.ibm.websphere.objectgrid.continuousquery

Class ContinuousQueryManagerFactory

java.lang.Object

com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryManagerFactory

---

```
public final class ContinuousQueryManagerFactory
extends Object
```

A factory class for obtaining the ContinuousQueryManager.

Since:

8.6, XC10 2.5

Constructor Summary

ContinuousQueryManagerFactory()

Method Summary

static ContinuousQueryManager	**getManager**(ObjectGrid objGrid) Factory that returns the single instance of a ContinuousQueryManager for the given object grid.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ContinuousQueryManagerFactory

public **ContinuousQueryManagerFactory**()

Method Detail

getManager

public static ContinuousQueryManager **getManager**(ObjectGrid objGrid)

Factory that returns the single instance of a ContinuousQueryManager for the given object grid.

Parameters:

`objGrid` - - the ObjectGrid this Continuous Query Manager will define continuous queries for.

Returns:

the ContinuousQueryManager instance.

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryManager

public interface **ContinuousQueryManager**

The management interface used to define new continuous queries, retrieve an existing query, or destroy a query. An instance can be retrieved via ContinuousQueryManagerFactory.getManager(com.ibm.websphere.objectgrid.ObjectGrid)

Since:

8.6, XC10 2.5

Method Summary

<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType>	**defineContinuousQuery**(String mapName, ContinuousQueryFilter filter, boolean keysOnly, boolean returnInitialResultSet, boolean notifyOnUpdated) Define a new continuous query.
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType>	**defineContinuousQuery**(String mapName, ContinuousQueryFilter filter, boolean keysOnly, boolean returnInitialResultSet, boolean notifyOnUpdated, Collection<ContinuousQueryListener<KeyType,ValueType>> continuousQueryListeners, boolean noCache, com.ibm.websphere.objectgrid.OutputFormat format, List<Integer> partitionSubset) Define a new continuous query.
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType>	**defineContinuousQuery**(String mapName, ContinuousQueryFilter filter, boolean keysOnly, boolean returnInitialResultSet, boolean notifyOnUpdated, List<Integer> partitionSubset) Define a new continuous query.
List<ContinuousQueryTopic<?,?>>	**getDefinedContinuousQueries**() Returns a list view of the currently defined Continuous Query Topics.
boolean	**removeContinuousQuery**(ContinuousQueryTopic<?,?> topic) Remove a defined continuous query.

Method Detail

defineContinuousQuery

<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType> **defineContinuousQuery**(String mapName,
ContinuousQueryFilter filter,
boolean keysOnly,
boolean returnInitialResultSet,
boolean notifyOn

```
Updated,
                                                                    Collection<Conti
nuousQueryListener<KeyType,ValueType>> continuousQueryListeners,
                                                                    boolean noCache,
                                                                    com.ibm.webspher
e.objectgrid.OutputFormat format,
                                                                    List<Integer> pa
rtitionSubset)
                                                        throws ContinuousQueryIncompatibleDu
plicateException,
                                                               UndefinedMapException
```

Define a new continuous query. This query will run in the grid container processes. When cache entries that match the filter criteria are inserted or deleted, the client's `ContinuousQueryCache` will be updated and any `ContinuousQueryListener` implementations will be invoked. This will also occur if a cache entry is updated such that it now matches, or no longer matches, the filter criteria. Optionally, notifications can be sent for update operations to cache entries that previously matched the criteria, and still match the criteria after the update.

Parameters:

`mapName` - - the name of the map the continuous query will be defined on.
`filter` - - the filter that the will determine the contents of the continuous query. Custom filter logic can be implemented by extending `AbstractCQFilter`.
`keysOnly` - - true if the query should only return the keys of items that satisfy the query; false to also send the values as well as the keys. If false, the `ContinuousQueryCache` and `ContinuousQueryNotificationEvent` objects will include the value associated with cache entries that match the query.
`returnInitialResultSet` - - true to run the filter on all existing matches in the map at the time of the query definition and return the items to the cache. If false, the client will only be notified of matches which occur after the query is defined. Returning the initial result set will have a high computational requirement for large maps.
`notifyOnUpdated` - - true causes this topic to get called every time an existing item that satisfies this query gets updated and continues to satisfy this query. This will drive `ContinuousQueryListener` call backs as well as updated values in the query cache if keysOnly is false.
`continuousQueryListeners` - - used to register listeners before the query is defined. These listeners will be guaranteed to be invoked for any entries in the initial result set, as well as changes to the result set which might occur before `ContinuousQueryTopic.addListener(ContinuousQueryListener)` can be invoked. Can be null.
`noCache` - - true to indicate that no caching of keys or values should occur on the client. Continuous Query listeners will still be notified.
`format` - - if `OutputFormat.RAW` is specified, keys and values returned from the `ContinuousQueryCache` and `ContinuousQueryListener` will be in `SerializedEntry` format when a `DataSerializer` is defined on the map. - if `OutputFormat.NATIVE` is specified, or when no `DataSerializer` is defined, keys and values will be returned as Objects.
`partitionSubset` - - Used to indicate that this continuous query should only be defined on the specified subset of partitions. Passing null or an empty list indicates this continuous query should be defined on all existing and future partitions containing this map. An IllegalArgumentException may be thrown if an invalid partition ID is passed.

Returns:

The `ContinuousQueryTopic` for this continuous query. If the call to this method would result in an topic which is identical to an existing topic being created, the existing instance is returned.

Throws:

`ContinuousQueryIncompatibleDuplicateException`
`UndefinedMapException`
`IllegalArgumentException`

---

defineContinuousQuery

```
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType> defineContinuousQuery(String mapName,
                                                                                   ContinuousQueryF
ilter filter,
                                                                                   boolean keysOnly
,
                                                                                   boolean returnIn
itialResultSet,
                                                                                   boolean notifyOn
Updated,
                                                                                   List<Integer> pa
rtitionSubset)
                                                                            throws ContinuousQueryIncompatibleDu
plicateException,
                                                                                   UndefinedMapException
```

Define a new continuous query. This query will run in the grid container processes. When cache entries that match the filter criteria are inserted or deleted, the client's `ContinuousQueryCache` will be updated and any `ContinuousQueryListener` implementations will be invoked. This will also occur if a cache entry is updated such that it now matches, or no longer matches, the filter criteria. Optionally, notifications can be sent for update operations to cache entries that previously matched the criteria, and still match the criteria after the update.

Parameters:

`mapName` - - the name of the map the continuous query will be defined on.
`filter` - - the filter that the will determine the contents of the continuous query. Custom filter logic can be implemented by extending `AbstractCQFilter`.
`keysOnly` - - true if the query should only return the keys of items that satisfy the query; false to also send the values as well as the keys. If false, the `ContinuousQueryCache` and `ContinuousQueryNotificationEvent` objects will include the value associated with cache entries that match the query.
`returnInitialResultSet` - - true to run the filter on all existing matches in the map at the time of the query definition and return the items to the cache. If false, the client will only be notified of matches which occur after the query is defined. Returning the initial result set will have a high computational requirement for large maps.
`notifyOnUpdated` - - true causes this topic to get called every time an existing item that satisfies this query gets updated and continues to satisfy this query. This will drive `ContinuousQueryListener` call backs as well as updated values in the query cache if keysOnly is false.
`partitionSubset` - - Used to indicate that this continuous query should only be defined on the specified subset of partitions. Passing null or an empty list indicates this continuous query should be defined on all existing and future partitions containing this map. An IllegalArgumentException may be thrown if an invalid partition ID is passed.

Returns:

The `ContinuousQueryTopic` for this continuous query. If the call to this method would result in an topic which is identical to an existing topic being created, the existing instance is returned.

Throws:

`ContinuousQueryIncompatibleDuplicateException`
`UndefinedMapException`
`IllegalArgumentException`

---

defineContinuousQuery

```
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType> defineContinuousQuery(String mapName,
                                                                                   ContinuousQueryF
ilter filter,
                                                                                   boolean keysOnly
,
                                                                                   boolean returnIn
itialResultSet,
                                                                                   boolean notifyOn
```

Updated)
throws ContinuousQueryIncompatibleDuplicateException,
UndefinedMapException

Define a new continuous query. This query will run in the grid container processes. When cache entries that match the filter criteria are inserted or deleted, the client's ContinuousQueryCache will be updated and any ContinuousQueryListener implementations will be invoked. This will also occur if a cache entry is updated such that it now matches, or no longer matches, the filter criteria. Optionally, notifications can be sent for update operations to cache entries that previously matched the criteria, and still match the criteria after the update. This query will be applied to all partitions.

Parameters:

`mapName` - - the name of the map the continuous query will be defined on.
`filter` - - the filter that the will determine the contents of the continuous query. Custom filter logic can be implemented by extending AbstractCQFilter.
`keysOnly` - - true if the query should only return the keys of items that satisfy the query; false to also send the values as well as the keys. If false, the ContinuousQueryCache and ContinuousQueryNotificationEvent objects will include the value associated with cache entries that match the query.
`returnInitialResultSet` - - true to run the filter on all existing matches in the map at the time of the query definition and return the items to the cache. If false, the client will only be notified of matches which occur after the query is defined. Returning the initial result set will have a high computational requirement for large maps.
`notifyOnUpdated` - - true causes this topic to get called every time an existing item that satisfies this query gets updated and continues to satisfy this query. This will drive ContinuousQueryListener call backs as well as updated values in the query cache if keysOnly is false.

Returns:

The ContinuousQueryTopic for this continuous query. If the call to this method would result in an topic which is identical to an existing topic being created, the existing instance is returned.

Throws:

ContinuousQueryIncompatibleDuplicateException
UndefinedMapException
IllegalArgumentException

---

removeContinuousQuery

boolean **removeContinuousQuery**(ContinuousQueryTopic<?,?> topic)

Remove a defined continuous query. Does the opposite of defineContinuousQuery(String, ContinuousQueryFilter, boolean, boolean, boolean). If the number of calls to this method for a given topic equals the number of calls to defineContinuousQuery that return the specified topic, the topic will cease receiving updates from the server, all listeners of this topic will be removed and the reference to this topic will be invalid.

Parameters:

`topic` - - the topic to remove

Returns:

true if the topic was removed, false otherwise

Throws:

IllegalArgumentException

---

getDefinedContinuousQueries

List<ContinuousQueryTopic<?,?>> **getDefinedContinuousQueries**()

Returns a list view of the currently defined Continuous Query Topics.

Returns:

a list of Continuous Query Topics

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryListener<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried

`ValueType` - Type of the value object for the map being queried

`public interface` **`ContinuousQueryListener<KeyType,ValueType>`**

This interface should be implemented to get a callback for a continuous query. When a Continuous Query listener is added to a Continuous Query Topic, that listener will be called after every change to the current set of results matching the continuous query.

Since:

8.6, XC10 2.5

Method Summary

void	**cacheUpdated**(ContinuousQueryNotificationEvent<KeyType,ValueType> event) This method will be called by every ContinuousQueryTopic that this listener is added to.

Method Detail

cacheUpdated

`void` **`cacheUpdated`**(ContinuousQueryNotificationEvent<KeyType,ValueType> event)

This method will be called by every ContinuousQueryTopic that this listener is added to. The caller will capture any Throwables thrown from calls to this method and create an FFDC entry. This ContinuousQueryListener will not be called again for the same ContinuousQueryNotificationEvent. Additionally, the performance of this ContinuousQueryListener may make an impact on the speed with which other ContinuousQueryListeners are called.

Parameters:

`event` - - ContinuousQueryNotificationEvent containing information about the change in the query result set

See Also:

ContinuousQueryTopic.addListener(ContinuousQueryListener), ContinuousQueryNotificationEvent

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryFilter<KeyType,ValueType,AttributeType,MatchType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried
`ValueType` - Type of the value object for the map being queried
`AttributeType` - Type of the attribute referenced by the attribute path
`MatchType` - Type of the object being compared to

All Superinterfaces:

Serializable

All Known Implementing Classes:

AbstractCQFilter, AndFilter, BinaryLogicalFilter, CompareFilter, EQFilter, FalseFilter, GTEFilter, GTFilter, IsNotNullFilter, IsNullFilter, LTEFilter, LTFilter, MatchFilter, NEQFilter, NotFilter, NotMatchFilter, OrFilter, TrueFilter

---

```
public interface ContinuousQueryFilter<KeyType,ValueType,AttributeType,MatchType>
extends Serializable
```

An interface which provides an abstraction to check if an object matches the criteria defined in a filter. Filters are invoked within the eXtreme Scale containers hosting partitions for the map on which the query is defined. All implementations must extend `AbstractCQFilter`.

Since:

8.6, XC10 2.5

Field Summary

static String	**POJO_ADDRESSABLEKEYNAME** The name used to identify the key objects in POJO maps.
static String	**POJO_PATHSEPARATOR** The path separator to use when identifying attribute paths for POJO maps.

boolean	**filter**(FilterContent<KeyType,ValueType> content) Checks if the supplied object passes the filter.

Field Detail

POJO_ADDRESSABLEKEYNAME

static final String **POJO_ADDRESSABLEKEYNAME**

The name used to identify the key objects in POJO maps. Not applicable to maps using XDF or a custom MapSerializedPlugin.

See Also:
CompareFilter.CompareFilter(String, Object), Constant Field Values

---

POJO_PATHSEPARATOR

static final String **POJO_PATHSEPARATOR**

The path separator to use when identifying attribute paths for POJO maps. Not applicable to maps using XDF or a custom MapSerializedPlugin.

See Also:
CompareFilter.CompareFilter(String, Object), Constant Field Values

Method Detail

filter

boolean **filter**(FilterContent<KeyType,ValueType> content)
throws ContinuousQueryException

Checks if the supplied object passes the filter.

Parameters:
content - A representation of the cache entry to be checked
Returns:
true if the object matches the filtering criteria, false otherwise
Throws:
ContinuousQueryException

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryCache<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried
`ValueType` - Type of the value object for the map being queried

`public interface` **`ContinuousQueryCache<KeyType,ValueType>`**

A Continuous Query Cache contains the keys and optionally the values that match a defined continuous query. The contents of this cache arrive asynchronously from the grid for which the query is defined. If the query is defined such that only the keys are stored in the continuous query cache then all operations that return a value will return null.

Since:
8.6, XC10 2.5

Method Summary

boolean	**containsKey**(KeyType key) Returns true if the cache contains the given key.
boolean	**containsValue**(ValueType value) Returns true if the cache contains the given value.
ValueType	**get**(KeyType key) Returns the value for the given key.
List<com.ibm.	

org.websphere.objectgrid.SessionHandle>	**getSessionHandles**(KeyType key) Returns SessionHandle objects that can be used to retrieve all query matches associated with this key.
boolean	**includesValues**() Returns true if this cache includes the values associated with the keys that have matched the query.
Set<KeyType>	**keySnapshot**() Returns a Set view of a snapshot of the keys contained in the cache.
int	**size**() Return the number of keys in the cache.

Method Detail

keySnapshot

Set<KeyType> **keySnapshot**()

Returns a Set view of a snapshot of the keys contained in the cache. The objects in the set are references to the keys in the query cache, therefore they should not be modified.

Returns:

a set view of the keys in the cache.

size

`int` **`size`**`()`

Return the number of keys in the cache.

Returns:
the number of keys in the cache.

---

get

`ValueType` **`get`**`(KeyType key)`

Returns the value for the given key. The object returned is a reference to the value in the query cache, therefore it should not be modified.

Parameters:
`key` - - the key whose associated value is to be returned
Returns:
the value in the cache for the specified key or null if either the cache does not contain this key or if the cache is configured to not contain values.

---

getSessionHandles

`List<com.ibm.websphere.objectgrid.SessionHandle>` **`getSessionHandles`**`(KeyType key)`

Returns `SessionHandle` objects that can be used to retrieve all query matches associated with this key. This method can only be invoked when using a PER_CONTAINER placement strategy for the map.

Parameters:
`key` - - the key for which SessionHandles will be returned
Returns:
a List of SessionHandles, one for each instance of this key in the grid.
See Also:
`MapSet.getPlacementStrategy()`

---

containsKey

`boolean` **`containsKey`**`(KeyType key)`

Returns true if the cache contains the given key.

Parameters:
`key` - - key to check for
Returns:
true if the specified key is in the cache at the time of the request.

---

containsValue

`boolean` **`containsValue`**`(ValueType value)`

Returns true if the cache contains the given value.

Parameters:
`value` - - value to check for
Returns:

true if the specified object is in the cache at the time of the request.

---

includesValues

`boolean **includesValues**()`

Returns true if this cache includes the values associated with the keys that have matched the query. If true, `containsValue(Object)` and `get(Object)` will return usable values, otherwise they will return null.

Returns:
true if this query cache contains values in addition to keys.

See Also:
`ContinuousQueryManager.defineContinuousQuery(java.lang.String, com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryFilter, boolean, boolean, boolean, java.util.Collection>, boolean, com.ibm.websphere.objectgrid.OutputFormat, java.util.List)`

Package com.ibm.websphere.objectgrid.management

This package contains the interfaces for all ObjectGrid MBeans.

See:

Description

Interface Summary

AgentManagerMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific Agent on a server process.
CatalogServiceManagementMBean	This MBean interface allows user to manipulate the behaviors of heartbeat and leader manager and other catalog service specific actions.
ContainerMBean	This MBean interface allows a client process to perform operations on and get status from an ObjectGrid container running in a dynamic environment.
DynamicServerMBean	This MBean interface allows a client process to access different attributes about a specific server process in a dynamic environment.
HashIndexMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific HashIndex on a server process.
MapMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific map on a server process.
ObjectGridMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific ObjectGrid on a server process.
PlacementMediationServiceMBean	This MBean interface allows a client process to perform operations on and get status from the PlacementMediationService running in a dynamic environment.
PlacementServiceMBean	This MBean interface allows a client process to perform operations on and get status from the PlacementService running in a dynamic environment.
QueryManagerMBean	This MBean interface allows a client process to perform operations on and get status from an ObjectGrid Query Manager running in a dynamic environment.
QuorumManagerMBean	Each catalog service has a QuorumManager and an associated MBean.
ServerMBean	This MBean interface allows a client process to access different attributes about a specific server process.
SessionMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific session.
ShardMBean	This MBean interface allows a client process to perform operations on and get status from a shard running in a dynamic environment.
ThreadPoolMBean	This MBean interface allows user to access the thread pool properties.

Package com.ibm.websphere.objectgrid.management Description

This package contains the interfaces for all ObjectGrid MBeans.

Overview

Each MBean interface has several methods to administer and monitor ObjectGrid services and components.

---

com.ibm.websphere.objectgrid.management

Interface ThreadPoolMBean

public interface **ThreadPoolMBean**

This MBean interface allows user to access the thread pool properties. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=ThreadPool
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
7.0.0.0 FIX2, XC10

Method Summary

int	**getActiveThreadCount**() Retrieves the approximate number of active threads in the pool.
int	**getMaximumSize**() Retrieves the maximum number of threads in the pool.
int	**getMinimumSize**() Retrieves the minimum number of threads in the pool.
String	**getName**() Retrieves the name of the ThreadPool.
void	**setMaximumSize**(int size) Sets the maximum thread pool size.
void	**setMinimumSize**(int size) Sets the minimum thread pool size.

Method Detail

setMaximumSize

`void` **`setMaximumSize`**`(int size)`

Sets the maximum thread pool size.

Parameters:
`size` - the maximum number of threads.

---

getMaximumSize

`int` **`getMaximumSize`**`()`

Retrieves the maximum number of threads in the pool.

Returns:
the maximum number of threads in the pool

---

setMinimumSize

`void` **`setMinimumSize`**`(int size)`

Sets the minimum thread pool size.

Parameters:
`size` - the minimum number of threads.

---

getMinimumSize

`int` **`getMinimumSize`**`()`

Retrieves the minimum number of threads in the pool.

Returns:
the minimum number of threads in the pool

---

getActiveThreadCount

`int` **`getActiveThreadCount`**`()`

Retrieves the approximate number of active threads in the pool.

Returns:
the number of active threads in the pool in use

---

getName

`String` **`getName`**`()`

Retrieves the name of the ThreadPool.

Returns:
the name of the ThreadPool

---

com.ibm.websphere.objectgrid.management

Interface ShardMBean

public interface **ShardMBean**

This MBean interface allows a client process to perform operations on and get status from a shard running in a dynamic environment. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=Shard,name=<objectgrid>,objectgrid=<objectgrid>,mapset=<mapset>
,partition=<partition id>,container=<container>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static String	**ROLE_SWAP_REQUESTED_WITH_SAME_TYPE** Indicates that this shard is the same type of shard as the requested swap type.
static String	**ROLE_SWAP_SUCCESSFUL** Indicates that the role swap was executed successfully.
static String	**ROLE_SWAP_TIMEOUT** Indicates that this shard has timed out waiting to inherit its requested role
st	

static String	**TYPE_INACTIVE** Indicates the shard type is the inactive role.
static String	**TYPE_PRIMARY** Indicates the shard type is the primary role.
static String	**TYPE_REPLICA_ASYNCHRONOUS** Indicates the shard type is the asynchronous replica role.
static String	**TYPE_REPLICA_SYNCHRONOUS** Indicates the shard type is the synchronous replica role.

Method Summary

long	**getActiveRequestCount**() Retrieves the number of requests currently being processed by this shard.
String	**getContainerName**() Retrieves the name of the container that is hosting this shard.
String	**getDomainName**() Retrieve the name of the catalog server grouping administering this shard.
long	**getForwardedRequestCount**() Retrieves the number of requests that this shard has forwarded since its inception.

String	**getMapSetName**() Retrieve the name of the MapSet in which the shard resides.
String	**getObjectGridName**() Retrieve the name of the ObjectGrid in which the shard resides.
String	**getPartitionName**() Retreive the name of the partition in which the shard resides.
long	**getProcessedRequestCount**() Retrieves the number of requests that this shard has processed since its inception.
String	**getState**() Retrieve the state of the shard.
long	**getTotalRequestCount**() Retrieves the number of requests that this shard has processed or forwarded since its inception.
String	**getType**() Retrieve the type of the shard.
String	**swapWithPrimary**() Causes this shard to swap roles with the primary shard for the partition.

Field Detail

ROLE_SWAP_SUCCESSFUL

```
static final String ROLE_SWAP_SUCCESSFUL
```

Indicates that the role swap was executed successfully.

Since:
7.1.0.0 FIX1

See Also:
swapWithPrimary(), Constant Field Values

ROLE_SWAP_REQUESTED_WITH_SAME_TYPE

`static final String` **`ROLE_SWAP_REQUESTED_WITH_SAME_TYPE`**

Indicates that this shard is the same type of shard as the requested swap type. No swap will be executed.

Since:
7.1.0.0 FIX1

See Also:
`swapWithPrimary()`, Constant Field Values

ROLE_SWAP_TIMEOUT

`static final String` **`ROLE_SWAP_TIMEOUT`**

Indicates that this shard has timed out waiting to inherit its requested role

Since:
7.1.0.0 FIX1

See Also:
`swapWithPrimary()`, Constant Field Values

TYPE_PRIMARY

`static final String` **`TYPE_PRIMARY`**

Indicates the shard type is the primary role. This means that this is the shard that handles all updates and coordinates state transitions with the replicas.

See Also:
Constant Field Values

TYPE_REPLICA_SYNCHRONOUS

`static final String` **`TYPE_REPLICA_SYNCHRONOUS`**

Indicates the shard type is the synchronous replica role. This means that this shard is receiving state updates from another shard that is acting as primary.

See Also:
Constant Field Values

TYPE_REPLICA_ASYNCHRONOUS

`static final String` **`TYPE_REPLICA_ASYNCHRONOUS`**

Indicates the shard type is the asynchronous replica role. This means that this shard is receiving state updates from another shard that is acting as primary.

See Also:
Constant Field Values

TYPE_INACTIVE

`static final String` **`TYPE_INACTIVE`**

Indicates the shard type is the inactive role. This means that this shard is not actively enrolled in the partition.

See Also:
Constant Field Values

Method Detail

getObjectGridName

`String` **`getObjectGridName`**`()`

Retrieve the name of the ObjectGrid in which the shard resides.

Returns:
The ObjectGrid name.

---

getMapSetName

`String` **`getMapSetName`**`()`

Retrieve the name of the MapSet in which the shard resides.

Returns:
The MapSet name.

---

getPartitionName

`String` **`getPartitionName`**`()`

Retreive the name of the partition in which the shard resides.

Returns:
The partition name.

---

getType

`String` **`getType`**`()`

Retrieve the type of the shard.

Returns:
The shard type.

---

getDomainName

`String` **`getDomainName`**`()`

Retrieve the name of the catalog server grouping administering this shard.

Returns:
The domain name.

---

getState

`String` **`getState`**`()`

Retrieve the state of the shard.

Returns:
The shard state.

---

getTotalRequestCount

`long` **`getTotalRequestCount`**`()`

Retrieves the number of requests that this shard has processed or forwarded since its inception.

Returns:
A count of the total number of requests.

---

getActiveRequestCount

`long` **`getActiveRequestCount`**`()`

Retrieves the number of requests currently being processed by this shard.

Returns:
A count of the active requests.

---

getForwardedRequestCount

`long` **`getForwardedRequestCount`**`()`

Retrieves the number of requests that this shard has forwarded since its inception.

Returns:
A count of the total number of forwarded requests.

---

getProcessedRequestCount

`long` **`getProcessedRequestCount`**`()`

Retrieves the number of requests that this shard has processed since its inception.

Returns:
A count of the total number of processed requests.

---

getContainerName

`String` **`getContainerName`**`()`

Retrieves the name of the container that is hosting this shard.

Returns:
The name of the container.
Since:
WAS XD 6.1.0.3

---

swapWithPrimary

`String` **`swapWithPrimary`**`()`

Causes this shard to swap roles with the primary shard for the partition. This shard becomes the primary while the shard that was previously the primary inherits this shard's former role.

If the role swap is not complete within 10 seconds, this operation will timeout.

Returns:

String the contains the return code of the operation

Since:

7.1.0.0 FIX1

See Also:

ROLE_SWAP_SUCCESSFUL, ROLE_SWAP_REQUESTED_WITH_SAME_TYPE, ROLE_SWAP_TIMEOUT

---

com.ibm.websphere.objectgrid.management

Interface SessionMBean

public interface **SessionMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific session. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=Session,name=<id>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
WAS XD 6.0.1, XC10

Method Summary

long	**getAccessedSessionsCount**() Gets acessed sessions count
long	**getAccessToNonExistentSessionCount**() Gets access to non-existent session count
long	**getActiveSessionsCount**() Gets active sessions count
long	**getAffinityBreaksCount**() Gets affinity breaks count
long	**getCacheDiscardsCount**() Gets cache discards count
long	**getCreatedSessionsCount**() Gets the map count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getInvalidatedByTimeoutCount**() Gets invalidated by timeout count
l	

long	**getInvalidatedSessionsCount**() Gets invalidated sessions count
long	**getMemoryCount**() Gets memory count
String	**getSessionID**() Gets the ID of the session instance associated with this MBean.
String	**getSessionStatsModule**() Gets a string representation of the `SessionStatsModule attributes loaded up by the retrieveStatsModule() method.`
com.ibm.websphere.objectgrid.stats.SessionStatsModule	**retrieveStatsModule**() Gets the `SessionStatsModule` used to retrieve statistics associated with the session for this MBean.

e	

Method Detail

retrieveStatsModule

`com.ibm.websphere.objectgrid.stats.SessionStatsModule` **`retrieveStatsModule`**`()`

Gets the `SessionStatsModule` used to retrieve statistics associated with the session for this MBean.

Returns:
an SessionStatsModule for statistics associated with this session

See Also:
`SessionStatsModule`

---

getSessionID

`String` **`getSessionID`**`()`

Gets the ID of the session instance associated with this MBean.

Returns:
the ID of the session instance associated with this MBean.

---

getSessionStatsModule

`String` **`getSessionStatsModule`**`()`

Gets a string representation of the `SessionStatsModule` attributes loaded up by the `retrieveStatsModule()` method.

Returns:
String form of SessionStatsModule

See Also:
`retrieveStatsModule()`, SessionStatsModule

---

getCreatedSessionsCount

`long` **`getCreatedSessionsCount`**`()`

Gets the map count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the number of entries in the map

See Also:
`retrieveStatsModule()`, MapStatsModule.getNumEntries(boolean)

---

getInvalidatedSessionsCount

`long` **`getInvalidatedSessionsCount`**`()`

Gets invalidated sessions count

Returns:
the count of invalidated Sessions

---

getActiveSessionsCount

long **getActiveSessionsCount**()

Gets active sessions count

Returns:
the count of active Sessions

---

getMemoryCount

long **getMemoryCount**()

Gets memory count

Returns:
memory count

---

getCacheDiscardsCount

long **getCacheDiscardsCount**()

Gets cache discards count

Returns:
cache discards count

---

getAffinityBreaksCount

long **getAffinityBreaksCount**()

Gets affinity breaks count

Returns:
affinity breaks count

---

getInvalidatedByTimeoutCount

long **getInvalidatedByTimeoutCount**()

Gets invalidated by timeout count

Returns:
count

---

getAccessToNonExistentSessionCount

long **getAccessToNonExistentSessionCount**()

Gets access to non-existent session count

Returns:
count

---

getAccessedSessionsCount

long **getAccessedSessionsCount**()

Gets acessed sessions count

Returns:
count

---

com.ibm.websphere.objectgrid.management

Interface ServerMBean

All Known Subinterfaces:

DynamicServerMBean

`public interface` **`ServerMBean`**

This MBean interface allows a client process to access different attributes about a specific server process.

Since:

WAS XD 6.0.1, XC10

Method Summary

String	**getServerName**() Gets the name of the server associated with this MBean.
void	**modifyServerTraceSpec**(String spec) **Deprecated.** *This is deprecated in version 7.1. See DynamicServerMBean.setTraceSpec(String)*
boolean	**stopServer**() Stops the server associated with this MBean.

Method Detail

getServerName

String **getServerName**()

Gets the name of the server associated with this MBean.

Returns:

the server name

---

stopServer

boolean **stopServer**()

Stops the server associated with this MBean.

Returns:

`true` if server was stopped, `false` if not

---

modifyServerTraceSpec

`void` **`modifyServerTraceSpec`**(`String` spec)

Deprecated. *This is deprecated in version 7.1. See* *`DynamicServerMBean.setTraceSpec(String)`*

Modifies the trace spec for the server associated with this MBean.

Parameters:

`spec` - new trace specification

---

com.ibm.websphere.objectgrid.management

Interface QuorumManagerMBean

public interface **QuorumManagerMBean**

Each catalog service has a QuorumManager and an associated MBean. The QuorumManager monitors and manages the quorum state of the catalog service grid. When quorum is enabled, the QuorumManager for each catalog service process detects when all catalog services in the grid have quorum or not. This MBean allows querying the current quorum state and allows administrators to force quorum when there is a network failure.

```
com.ibm.websphere.objectgrid:type=QuorumManager
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

The following notifications are available:

com.ibm.websphere.objectgrid.quorum.lost

Description:	Catalog service quorum has been lost.
Message:	A translated string identifying the number of active catalog servers and the number in quorum.

com.ibm.websphere.objectgrid.quorum.changed

Description:	The catalog service has quorum, but the number of catalog servers required for quorum has changed.
Message:	A translated string identifying the number of active catalog servers and the number in quorum.

Since:

7.0, XC10

Field Summary

static String	QUORUM_CHANGED_NOTIFICATION
stat	

static [String](#)	**QUORUM_LOST_NOTIFICATION**

Method Summary

[String](#)[]	**getActiveCatalogServerNames**() Retrieves the names of the known active catalog service processes.
int	**getActiveCatalogServers**() Retrieve the known number of active catalog service processes.
int	**getQuorumCatalogServers**() Retrieve the number of catalog service processes required for quorum.
void	**overrideQuorum**() This operation forces surviving catalog service grid processes to reestablish a quorum.

Field Detail

QUORUM_LOST_NOTIFICATION

static final [String](#) **QUORUM_LOST_NOTIFICATION**

See Also:
[Constant Field Values](#)

QUORUM_CHANGED_NOTIFICATION

static final [String](#) **QUORUM_CHANGED_NOTIFICATION**

See Also:
[Constant Field Values](#)

Method Detail

overrideQuorum

void **overrideQuorum**()
throws [Exception](#)

This operation forces surviving catalog service grid processes to reestablish a quorum.

If a portion the catalog service grid fails or is divided due to a network failure, the grid will lose quorum. Once the administrator identifies the failure and the viable portion of the grid, this operation can be invoked on any of the surviving catalog service processes to reestablish a quorum. Reestablishing a quorum will allow the catalog service to

continue to react to failures and topology changes.

Throws:
Exception

---

getActiveCatalogServers

`int` **`getActiveCatalogServers`**`()`

Retrieve the known number of active catalog service processes.

Returns:
the known number of active catalog service processes.

---

getQuorumCatalogServers

`int` **`getQuorumCatalogServers`**`()`

Retrieve the number of catalog service processes required for quorum.

Returns:
the number of catalog service processes required for quorum.

---

getActiveCatalogServerNames

`String[]` **`getActiveCatalogServerNames`**`()`

Retrieves the names of the known active catalog service processes.

Returns:
the names of the known active catalog service processes.
Since:
7.1

---

com.ibm.websphere.objectgrid.management

Interface QueryManagerMBean

public interface **QueryManagerMBean**

This MBean interface allows a client process to perform operations on and get status from an ObjectGrid Query Manager running in a dynamic environment. The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=QueryManager,name=<grid name>,mapset=<mapset name>,partition=<p
artition number>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1.0.5, XC10

Method Summary

double	**getPlanCreationTime**(String query) Gets the query's plan creation time attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryExecutionCount**(String query) Gets the query's execution count attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryExecutionTime**(String query) Gets the query's execution time attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryFailureCount**(String query) Gets the query's failure count attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryResultCount**(String query) Gets the query's result count attribute loaded up by the `retrieveStatsModule()` method.

com.ibm.websphere.objectgrid.stats.QueryStatsModule	
	retrieveStatsModule(String query) Gets the QueryStatsModule used to retrieve statistics associated with the specified query String

Method Detail

getPlanCreationTime

double **getPlanCreationTime**(String query)

Gets the query's plan creation time attribute loaded up by the retrieveStatsModule() method.

Returns:
the plan creation time for this query in milliseconds

See Also:
retrieveStatsModule(String), QueryStatsModule.getPlanCreationTime(boolean copy)

---

getQueryExecutionTime

double **getQueryExecutionTime**(String query)

Gets the query's execution time attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the execution time for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryExecutionTime(boolean copy)`

---

getQueryExecutionCount

`double` **`getQueryExecutionCount`**`(String query)`

Gets the query's execution count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the execution count for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryExecutionCount(boolean copy)`

---

getQueryResultCount

`double` **`getQueryResultCount`**`(String query)`

Gets the query's result count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the result count for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryResultCount(boolean copy)`

---

getQueryFailureCount

`double` **`getQueryFailureCount`**`(String query)`

Gets the query's failure count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the failure count for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryFailureCount(boolean copy)`

---

retrieveStatsModule

`com.ibm.websphere.objectgrid.stats.QueryStatsModule` **`retrieveStatsModule`**`(String query)`

Gets the `QueryStatsModule` used to retrieve statistics associated with the specified query String

Returns:
an QueryStatsModule for statistics associated with the specified query String

---

com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

All Superinterfaces:

com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean

---

```
public interface PlacementServiceMBean
extends com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean
```

This MBean interface allows a client process to perform operations on and get status from the PlacementService running in a dynamic environment. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=PlacementService
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static int	**ALL** Constant representing a all shard types
static int	**ASYNCHRONOUS_REPLICA** Constant representing an asynchronous replica shard type.
static int	**PRIMARY** Constant representing a primary shard type.
sta	

static int	**SYNCHRONOUS_REPLICA** Constant representing a synchronous replica shard type.

Fields inherited from interface com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean
HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE, HEARTBEAT_FREQUENCY_LEVEL_RELAXED, HEARTBEAT_FREQUENCY_LEVEL_TYPICAL

Method Summary

String	**balanceShardTypes**(String objectGridName, String mapSetName) The placement service will examine the distribution of primaries and replicas for a given mapSet and attempt (if zone rules and other balancing constraints allow) to achieve a consistent primary to replica ratio across the set of containers.
String	**balanceStatus**(String objectGridName, String mapSetName) Check the balance status (suspended or resumed) for a specified MapSet.
String	**collectContainerStatus**(String objectGridName, String mapSetName) Retrieves the container status for all containers in the domain.
boolean	**enableForPlacement**(String containerName) Re-enables a disabled container for placement.
String	**getCoreGroups**() Gets the coregroup status.
TabularData	**getDisabledForPlacement**() Retrieves TabularData of containers which have been disabled because of the failure of shard placement operations.
boolean	**getInitialized**() Returns whether this instance of the Placement Service finished initialization.

String	**getObjectGridNames**() Gets the names of all ObjectGrids and their mapsets in the domain.
String	**listCoreGroupMembers**(String coreGroupName) List the coregroup members for a given coregroup.
String	**listObjectGridPlacement**(String objectGridName, String mapSetName) List the placement of shards for each container in the domain.
String	**listObjectGridPlacementStatus**(String objectGridName, String mapSetName) List the current placement status.
String	**listPartition**(String objectGridName, String mapSetName, String partitionId) List the partition placement status in the domain.
String	**listShards**(String objectGridName, String mapSetName, String containerName, int mask) List the shard placement status.
String	**listVerifiedRoutingTable**(String objectGridName) **Deprecated.**
String	**replaceLostShards**(String objectGridName, String mapSetName) Lost shards are placed onto the UNREPAIRED container when autoReplaceLostShards is disabled.
String	**resumeBalancing**(String objectGridName, String mapSetName) Execute balancing operation at next opportunity and allow execution of future balancing attempts for the map set specified.
List	**retrieveAllServersJMXAddresses**() Retrieves a List of JMX addresses for all servers that have registered with the placement service.
Str	**retrieveMapSetName**(String gridName, String mapName)

[String](#)	Retrieves the name of the MapSet in which the specified map is defined.
[List](#)	**retrieveServerJMXAddress**([String](#) hostName, [String](#) serverName) Retrieves a List of JMX address strings for a specific host and server name that has registered with the placement service.
[TabularData](#)	**retrieveServerJMXAddressesWithInfo**([String](#) hostName, [String](#) serverName) Retrieves TabularData of JMX address strings for a specific host and server name that has registered with the placement service.
[String](#)	**suspendBalancing**([String](#) objectGridName, [String](#) mapSetName) Prevent future balancing attempts for a specific map set.
[String](#)	**tearDownServers**([String](#)[] servers) Each of the container servers that are passed into this method will be stopped.
[String](#)	**triggerPlacement**([String](#) objectGridName, [String](#) mapSetName) Placement normally occurs implicitly after an event such as an ObjectGrid container starting or stopping.

Methods inherited from interface com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean
getHeartBeatFrequencyLevel, setHeartBeatFrequencyLevel

Field Detail

PRIMARY

static final int **PRIMARY**

Constant representing a primary shard type.

See Also:
[Constant Field Values](#)

SYNCHRONOUS_REPLICA

static final int **SYNCHRONOUS_REPLICA**

Constant representing a synchronous replica shard type.

See Also:
Constant Field Values

ASYNCHRONOUS_REPLICA

```
static final int ASYNCHRONOUS_REPLICA
```

Constant representing an asynchronous replica shard type.

See Also:
Constant Field Values

ALL

```
static final int ALL
```

Constant representing a all shard types

See Also:
Constant Field Values

Method Detail

retrieveServerJMXAddress

```
List retrieveServerJMXAddress(String hostName,
                              String serverName)
```

Retrieves a List of JMX address strings for a specific host and server name that has registered with the placement service.

Parameters:
`hostName` - The name of the host to retrieve the JMX addresses.
`serverName` - The name of the server to retrieve the JMX addresses.
Returns:
the List of all JMX address strings for the specified host and server name.

---

retrieveServerJMXAddressesWithInfo

```
TabularData retrieveServerJMXAddressesWithInfo(String hostName,
                                               String serverName)
```

Retrieves TabularData of JMX address strings for a specific host and server name that has registered with the placement service. Null host or null server names can be used to retrieve several results. Using a null host and null server name will retrieve all of the servers. The TabularData contains CompositeData with the following items, where each CompositeData represents a server:

Item Name	Type	Description
JMXServiceURL	String	JMX Service URL
HostName	String	Host name
ServerName	String	Server Name

Parameters:
`hostName` - The name of the host for which to retrieve JMX addresses. Use null or an empty string to retrieve JMX addresses for any host.
`serverName` - The name of the server for which to retrieve JMX addresses. Use null or an empty string to retrieve JMX addresses for any server.

Returns:
the TabularData of all JMX address strings for the specified host and server name.
Since:
8.5

---

retrieveAllServersJMXAddresses

`List` **`retrieveAllServersJMXAddresses`**`()`

Retrieves a List of JMX addresses for all servers that have registered with the placement service.

Returns:
the List of all servers' JMX addresses

---

collectContainerStatus

`String` **`collectContainerStatus`**`(String objectGridName,`
`String mapSetName)`

Retrieves the container status for all containers in the domain.

The results are returned in the following format:

```
<container name="<container>" zoneName="<zone>" hostName="<host>" serverName="<server>">
  <shard type="<type>" partitionName="<partition>"/>
</container>
```

Parameters:
`objectGridName` - The name of the ObjectGrid for which to get container status.
`mapSetName` - The name of the mapset for which to get the container status.
Returns:
The String status object for all containers in XML form.

---

listObjectGridPlacement

`String` **`listObjectGridPlacement`**`(String objectGridName,`
`String mapSetName)`

List the placement of shards for each container in the domain.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset">
  <container name="<container>" zoneName="<zone>" hostName="<host>" serverName="<server>">
    <shard type="<type>" partitionName="<partition>" reserved="<true>"/>
  /container>
</objectGrid>
```

NOTE: The default value for the "reserved" attribute is false.

Parameters:
`objectGridName` - The name of the ObjectGrid for which to get placement status.
`mapSetName` - The name of the mapset for which to get the placement status.
Returns:
The placement status in XML form.

---

listObjectGridPlacementStatus

```
String listObjectGridPlacementStatus(String objectGridName,
                                      String mapSetName)
```

List the current placement status.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <configuration>
    <attribute name="<placementStrategy>" value="<strategy>"/>
    <attribute name="<numInitialContainers>" value="<num>"/>
    <attribute name="<minSyncReplicas>" value="<min>"/>
    <attribute name="<developmentMode>" value="<mode>"/>
  </configuration>
  <runtime>
    <attribute name="<numContainers>" value="<num>"/>
    <attribute name="<numMachines>" value="<num>"/>
    <attribute name="<numOutstandingWorkItems>" value="<num>"/>
    <attribute name="<numActiveZones>" value="<num>"/>
  </runtime>
</objectGrid>
```

Parameters:

`objectGridName` - The name of the ObjectGrid for which to get placement status.

`mapSetName` - The name of the mapset for which to get the placement status.

Returns:

The placement status in XML form.

---

getCoreGroups

```
String getCoreGroups()
```

Gets the coregroup status.

The results are returned in the following format:

```
<coreGroup name="<coregroup>">
  <coreGroupLeader hostName="<host>" serverName="<server>"/>
  <coreGroupMember hostName="<host>" serverName="<server>"/>
</coreGroup>
```

Returns:

the coregroup status in XML form.

---

listCoreGroupMembers

```
String listCoreGroupMembers(String coreGroupName)
```

List the coregroup members for a given coregroup.

The results are returned in the following format:

```
<coreGroup name="<coregroup>">
  <coreGroupMember hostName="<host>" serverName="<server>"/>
</coreGroup>
```

Parameters:

`coreGroupName` - The name of the coregroup for which to get the members.

Returns:

The coregroup members in XML form.

---

listPartition

```
String listPartition(String objectGridName,
                     String mapSetName,
                     String partitionId)
```

List the partition placement status in the domain. The results are returned in the following format:

```
<partition name="<partition>">
  <shard type="<type>" containerName="<container>" hostName="<host>" serverName="<server>"/>
</partition>
```

Parameters:

`objectGridName` - The name of the ObjectGrid for which to get placement status.
`mapSetName` - The name of the mapset for which to get the placement status.
`partitionId` - The name of the partition for which to get the placement status.

Returns:

The partition placement status in the XML form.

---

listShards

```
String listShards(String objectGridName,
                  String mapSetName,
                  String containerName,
                  int mask)
```

List the shard placement status.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <container name="<container>" zoneName="<zone>" hostName="<host>" serverName="<server>">
    <shard type="<type>" partitionName="<partition>"/>
  </container>
</objectGrid>
```

Parameters:

`objectGridName` - The name of the ObjectGrid for which to get placement status.
`mapSetName` - The name of the mapset for which to get the placement status.
`containerName` - The name of the container for which to get the placement status. If empty string (""), get shard placement for all containers.
`mask` - The Integer mask to determine for which shard types to get status.

Returns:

The shard placement status in XML form.

See Also:

ALL, PRIMARY, SYNCHRONOUS_REPLICA, ASYNCHRONOUS_REPLICA

---

getObjectGridNames

```
String getObjectGridNames()
```

Gets the names of all ObjectGrids and their mapsets in the domain.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>"/>
```

Returns:

the names of all ObjectGrids and their mapsets in the domain in XML form.

---

replaceLostShards

```
String replaceLostShards(String objectGridName,
                         String mapSetName)
```

Lost shards are placed onto the UNREPAIRED container when autoReplaceLostShards is disabled. Shards on the UNREPAIRED will not be placed until this method is called.

Calling this method will move shards off the UNREPAIRED container onto the UNASSIGNED container.

Balance and placement operations will be queued up for the MapSet specified. These operations will execute when all outstanding placement work from previous events has completed.

The string returned is an XML representation of the shards that moved as a result of the call to this method.

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <shard type="<type>" partitionName="<partition>">
     <currentContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<se
rver>"/>
     <previousContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<s
erver>"/>
   </shard>
 </objectGrid>
```

The returned XML will look as follows when no shards have been moved:

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <!-- No shards were moved -->
 </objectGrid>
```

Parameters:

`objectGridName` - replace lost shards for this ObjectGrid

`mapSetName` - replace lost shards for this MapSet

Returns:

An XML String containing shards that have moved

Since:

WAS XD 6.1.0.5

---

triggerPlacement

```
String triggerPlacement(String objectGridName,
                        String mapSetName)
```

Placement normally occurs implicitly after an event such as an ObjectGrid container starting or stopping.

Calling this method will trigger a placement operation for the ObjectGrid and MapSet specified.

Under normal circumstances, the `numInitialContainers` attribute (in the deployment policy) must be met in order for placement to occur. However, when this method is called, the numInitialContainers value is ignored.

The string returned is an XML representation of the shards that moved as a result of the call to this method.

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <shard type="<type>" partitionName="<partition>">
     <currentContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<se
rver>"/>
     <previousContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<s
erver>"/>
   </shard>
 </objectGrid>
```

The returned XML will look as follows when no shards have been moved:

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <!-- No shards were moved -->
 </objectGrid>
```

Parameters:

`objectGridName` - trigger placement for this ObjectGrid
`mapSetName` - trigger placement for this MapSet

Returns:

An XML String containing shards that have moved

Since:

WAS XD 6.1.0.5

See Also:

`ObjectGridDeployment.addMapSet(com.ibm.websphere.objectgrid.deployment.MapSet)`, `MapSet.setNumInitialContainers(int)`

---

tearDownServers

`String` **tearDownServers**(`String`[] servers)

Each of the container servers that are passed into this method will be stopped. If the server cannot be reached, all of the server's artifacts will be removed.

Use this method if servers are found to be in a corrupt state or bindings need to be cleared from the catalog server.

The string returned is an XML representation of the results of the attempt to tear down each of the servers. If the command is successful, the XML will look as follows:

```
<domain name="<domain>">
  <server name="<server>" tearDownSuccessful="true"/>
  <server name="<server>" tearDownSuccessful="true"/>
</domain>
```

If the command is not successful, the string will look as follows (where the exception element is only present if an exception is part of the failure):

```
<domain name="<domain>">
  <server name="<server>" tearDownSuccessful="false" reason="<String>">
     <exception type="<String>" message="<String>" stack="<String>"/>
  </server>
</domain>
```

Parameters:

`servers` - String array of servers to tear down.

Returns:

An XML String containing the results of tear down attempts.

Since:

WAS XD 6.1.0.5 FIX2

---

listVerifiedRoutingTable

```
@Deprecated
String listVerifiedRoutingTable(String objectGridName)
```

Deprecated.

This method is deprecated. The com.ibm.websphere.objectgrid.client.RouteTableValidation utility replaces this method.

Calling this method will return an XML string of the current known routing table. The Placement service will contact each shard and return state on whether it was able to verify that's shard's existence. All shards will be included in the XML doc, whether they were reachable or not. The user can use the `reachable` attribute below to filter valid or invalid shards.

```
 <objectGrid name="<objectgrid>" name="<name>">
   <primary zone="<zone>"> partition="<partition>"> state="<reachable>"> ipaddress="<ipaddres
s>">
     <replica zone="<zone>"> partition="<partition>"> state="<reachable>"> ipaddress="<ipaddr
ess>">
   </primary>
 </objectGrid>
```

Parameters:

`objectGridName` - retrieve routing table for this ObjectGrid

Returns:

An XML String containing a pre-verified routing table

Since:

WAS XD 6.1.0.5 FIX2

---

retrieveMapSetName

```
String retrieveMapSetName(String gridName,
                          String mapName)
```

Retrieves the name of the MapSet in which the specified map is defined.

Parameters:

`gridName` - the name of the ObjectGrid

`mapName` - the name of the map

Returns:

the name of the MapSet in which the specified map is defined.

Since:

7.0

---

getInitialized

```
boolean getInitialized()
```

Returns whether this instance of the Placement Service finished initialization.

Returns:

true if initialized, false if not initialized.

Since:

8.5

---

balanceShardTypes

```
String balanceShardTypes(String objectGridName,
                         String mapSetName)
```

The placement service will examine the distribution of primaries and replicas for a given mapSet and attempt (if zone rules and other balancing constraints allow) to achieve a consistent primary to replica ratio across the set of containers.

If the number of primaries or the number of replicas do not divide evenly across the containers, some tolerance must be allowed for the ratio to differ from container to container. However, the difference in the number of primaries from one container to the next will not be greater than 1. Similarly, the difference in the number of replicas from one container to the next will not be greater than 1.

Null arguments are not allowed as input to this method.

The results are returned in the following format:

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <shard type="<type>" partitionName="<partition>">
     <currentContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<se
rver>"/>
     <previousContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<s
erver>"/>
   </shard>
 </objectGrid>
```

If no shards were moved or a problem was encountered attempting to execute this method, no shard elements will appear in the XML output. A detail element will appear instead. The message attribute will have further information.

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <detail message="<message>" />
 </objectGrid>
```

Parameters:

`objectGridName` - the grid

`mapSetName` - the map set within the grid

Returns:

An XML String containing the results of the attempt to redistribute shards for better primary/replica balance

Since:

7.1.1

---

suspendBalancing

```
String suspendBalancing(String objectGridName,
                        String mapSetName)
```

Prevent future balancing attempts for a specific map set. Balancing work that is in progress will be allowed to complete.

Other placement activities are allowed to execute while balancing is suspended.

- shard promotion due to container loss
- shard role swap
- shard reservation
- triggerPlacement
- replaceLostShards

Balancing will remain suspended until it is resumed by calling `resumeBalancing(String, String)`.

Null arguments are not allowed as input to this method.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>"/>
</objectGrid>
```

Additionally, an optional detail element may be contained within the suspendBalancing element. The detail element will include additional data regarding execution of this method. The XML result will be in the following format when a detail element is included:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>">
    <detail message="<message>" />
  <suspendBalancing/>
</objectGrid>
```

Parameters:

`objectGridName` - suspend balancing for the map set specified within this ObjectGrid
`mapSetName` - suspend balancing for this map set

Returns:

An XML String containing the results of the attempt to suspend balancing

Since:

7.1.0.3

See Also:

`resumeBalancing(String, String)`

---

resumeBalancing

```
String resumeBalancing(String objectGridName,
                       String mapSetName)
```

Execute balancing operation at next opportunity and allow execution of future balancing attempts for the map set specified. Balancing is executed in reaction to key placement events. Such events include containers starting and containers stopping.

By default, balancing work is executed unless `suspendBalancing(String, String)` has been called for the map set.

Null arguments are not allowed as input to this method.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>"/>
</objectGrid>
```

Additionally, an optional detail element may be contained within the suspendBalancing element. The detail element will include additional data regarding execution of this method. The XML result will be in the following format when a detail element is included:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>">
    <detail message="<message>" />
  <suspendBalancing/>
</objectGrid>
```

Parameters:

`objectGridName` - resume balancing for the map set specified within this ObjectGrid

`mapSetName` - resume balancing for this map set

Returns:

An XML String containing the results of the attempt to resume balancing

Since:

7.1.0.3

See Also:

suspendBalancing(String, String)

---

balanceStatus

```
String balanceStatus(String objectGridName,
                     String mapSetName)
```

Check the balance status (suspended or resumed) for a specified MapSet.

Null arguments are not allowed as input to this method.

The string returned is an XML representation of the balance status. The XML will look as follows:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <balanceStatus suspended="<suspended>" />
</objectGrid>
```

Additionally, an optional detail element may be contained within the balanceStatus element. When balancing has been pre-suspended, the message attribute of the detail element will contain the following message.

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <balanceStatus suspended="true" >
    <detail message="Balancing has been pre-suspended for this mapSet." />
  </balanceStatus>
</objectGrid>
```

Parameters:

`objectGridName` - check balance status for the map set specified within this ObjectGrid

`mapSetName` - check balance status for this map set

Returns:

An XML String containing the balance status

Since:

7.1.1

See Also:

suspendBalancing(String, String), resumeBalancing(String, String)

---

enableForPlacement

```
boolean enableForPlacement(String containerName)
```

Re-enables a disabled container for placement. A container may become disabled because of a falure to place a shard into into the container.

Use the getDisabledForPlacement() attribute to determine which containers are disabled.

Parameters:

`containerName` - The name of the container to re-enable.

Returns:

Answers true if the container's status was changed from disabled to enabled, false if

the container was already enabled for placement.

Since:

8.6, XC10 2.5

---

getDisabledForPlacement

[TabularData](#) **getDisabledForPlacement**()

Retrieves TabularData of containers which have been disabled because of the failure of shard placement operations. The TabularData contains CompositeData with the following items, where each CompositeData represents a container:

Item Name	Type	Description
Container	String	A container that has been disabled for placement

Returns:

A TabularData of the names of disabled containers.

Since:

8.6, XC10 2.5

---

com.ibm.websphere.objectgrid.management

Interface PlacementMediationServiceMBean

public interface **PlacementMediationServiceMBean**

This MBean interface allows a client process to perform operations on and get status from the PlacementMediationService running in a dynamic environment. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=PlacementMediationService
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
7.1, XC10

Method Summary

CompositeData	**dismissLink**(String foreignDomain) Dismiss a previously established link with the foreign domain specified.
CompositeData	**establishLink**(String foreignDomain, String endPoints) Establish a link between this domain and the foreign domain specified.
TabularData	**getLinkedDomains**() Retrieve the foreign domains that have an active link with the local domain.

[a](#)	
[Tabular Data](#)	**[getLinkedDomainsWithGrids](#)**() Retrieve the foreign domains that have an active link with the local domain and the map sets eligible for linking.

Method Detail

establishLink

[CompositeData](#) **establishLink**([String](#) foreignDomain,
[String](#) endPoints)

Establish a link between this domain and the foreign domain specified. This is functionally equivalent to providing the foreign domain and its end points in the server properties file at server startup time.

Domains that are linked will share placement with each other. When compatible map sets are detected within linked domains, a multi-primary topology will be achieved. Data written to a primary is either domain will be asynchronously replicated to the other domain.

The result is a CompositeData that includes the following items:

Item Name	Type	Description
Result	String	The result of the attempt.
StatusBefore	String	The status of the link before the attempt was made to establish the link.
StatusAfter	String	The status of the link after the attempt was made to establish the link.

Parameters:

`foreignDomain` - the name of the foreign domain
`endPoints` - end points of the foreign domain

Returns:

CompositeData representing the status of the attempt to link with the foreign domain

See Also:

`CatalogServerProperties.setForeignDomains(String)`,
`CatalogServerProperties.setDomainEndPoints(String, String)`

---

dismissLink

[CompositeData](#) **dismissLink**([String](#) foreignDomain)

Dismiss a previously established link with the foreign domain specified. Any map sets that were participating in a multi-primary topology will be disconnected from each other. Data will no longer be replicated from between domains.

The result is a CompositeData that includes the following items:

Item Name	Type	Description
Result	String	The result of the attempt. Can be one of: SUCCESS, FAILURE, NOP
Status Before	String	The status of the link before the attempt was made to dismiss the link. Can be one of: LINKED, ESTABLISHING_LINK, UNLINKED, DISMISSING_LINK
Status After	String	The status of the link after the attempt was made to dismiss the link. Can be one of: LINKED, ESTABLISHING_LINK, UNLINKED, DISMISSING_LINK

Parameters:

`foreignDomain` - the name of the foreign domain

Returns:

CompositeData representing the status of the attempt to dismiss the link with the foreign domain

---

getLinkedDomains

TabularData **getLinkedDomains**()

Retrieve the foreign domains that have an active link with the local domain.

The result is a TabularData where each row is a CompositeData that includes the following items:

Item Name	Type	Description
Domain	String	The name of the foreign domain linked to the local domain.

Returns:

TabularData representing the foreign domains linked to this domain

---

getLinkedDomainsWithGrids

TabularData **getLinkedDomainsWithGrids**()

Retrieve the foreign domains that have an active link with the local domain and the map sets eligible for linking.

The result is a TabularData where each row is a CompositeData that includes the following items:

Item Name	Type	Description
Domain	String	The name of the foreign domain linked to the local domain.
ObjectGrid	String	The name of the ObjectGrid that is compatible with the foreign domain.
MapSet	String	The name of the map set that is compatible with the foreign domain.

Returns:

TabularData representing the foreign domains and eligible map sets linked to this domain

Since:

8.6, XC10 2.5

com.ibm.websphere.objectgrid.management

Interface ObjectGridMBean

public interface **ObjectGridMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific ObjectGrid on a server process. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGrid,name=<objectgrid>,mapset=<mapset>,partition=<partition id>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
WAS XD 6.0.1, XC10

Method Summary

String	**getContainerName**() Gets the name of the container name containing the replication group member for the ObjectGrid associated with this MBean.
long	**getCurrentRevision**() Retrieves the current revision number of this ObjectGrid shard.
String	**getDomainName**() Retrieves the domain name of this ObjectGrid shard.
TabularData	**getKnownRevisions**() An ObjectGrid shard may exist over several different lifetimes.
String	**getLifetimeId**() Retrieves the lifetime id for this ObjectGrid shard.

[ng](#)	
[String](#)	**getObjectGridName**() Gets the name of the ObjectGrid associated with this MBean.
long	**getOGCount**() Gets the ObjectGrid count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getOGMaxTranTime**() Gets the maximum transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getOGMeanTranTime**() Gets the mean transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getOGMinTranTime**() Gets the minimum transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
[String](#)	**getOGStatsModule**() Gets a string representation of the `OGStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`
long	**getOGTotalTranTime**() Gets the total transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getOGTransPerSecond**() Transactions per second attribute loaded up by the retrieveStatsModule call.
[TabularData](#)	**getPrimaryShardLinks**() Get the shard's list of foreign or domestic linked primaries.
[String](#)	**getServerName**() Gets the name of the server containing the replication group member for the ObjectGrid associated with this MBean.
com.	

com.ibm.websphere.objectgrid.stats.OGStatsModule
retrieveStatsModule() Gets the OGStatsModule used to retrieve statistics associated with the ObjectGrid for this MBean.

Method Detail

retrieveStatsModule

com.ibm.websphere.objectgrid.stats.OGStatsModule **retrieveStatsModule**()

Gets the OGStatsModule used to retrieve statistics associated with the ObjectGrid for this MBean.

Returns:
an OGStatsModule for statistics associated with this ObjectGrid

See Also:
OGStatsModule

---

getObjectGridName

String **getObjectGridName**()

Gets the name of the ObjectGrid associated with this MBean.

Returns:
name of the ObjectGrid

---

getServerName

String **getServerName**()

Gets the name of the server containing the replication group member for the ObjectGrid associated with this MBean.

Returns:
the name of server containing the replication group member for the ObjectGrid associated with this MBean.

---

getContainerName

String **getContainerName**()

Gets the name of the container name containing the replication group member for the ObjectGrid associated with this MBean.

Returns:
the name of container containing the replication group member for the ObjectGrid associated with this MBean.
Since:
8.5

---

getOGStatsModule

String **getOGStatsModule**()

Gets a string representation of the `OGStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`

Returns:
String form of OGStatsModule
See Also:
retrieveStatsModule(), OGStatsModule

---

getOGCount

long **getOGCount**()

Gets the ObjectGrid count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the number of transactions
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGMaxTranTime

long **getOGMaxTranTime**()

Gets the maximum transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the maximum transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGMinTranTime

long **getOGMinTranTime**()

Gets the minimum transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the minimum transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGMeanTranTime

double **getOGMeanTranTime**()

Gets the mean transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the mean transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGTotalTranTime

long **getOGTotalTranTime**()

Gets the total transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the total transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGTransPerSecond

long **getOGTransPerSecond**()

Transactions per second attribute loaded up by the retrieveStatsModule call. retrieveStatsModule() or refreshStatsModule() method.

Returns:
the transactions per second for the ObjectGrid
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getCurrentRevision

long **getCurrentRevision**()

Retrieves the current revision number of this ObjectGrid shard.

Returns:
the current revision number of this ObjectGrid shard.
Since:
7.1

---

getDomainName

String **getDomainName**()

Retrieves the domain name of this ObjectGrid shard.

Returns:
the name of the domain name of this ObjectGrid shard.
Since:
7.1

---

getLifetimeId

String **getLifetimeId**()

Retrieves the lifetime id for this ObjectGrid shard.

Returns:
the lifetime id for this ObjectGrid shard.
Since:
7.1

---

getKnownRevisions

TabularData **getKnownRevisions**()

An ObjectGrid shard may exist over several different lifetimes. As such, each shard instance will have a unique lifetime id and revision number associated with it. This method returns a TabularData object representing the known history of revision numbers for each lifetime. Each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Domain	String	The domain name of this ObjectGrid shard.	7.1
Server	String	The name of the server owning the lifetime id.	8.5
LifetimeId	String	The lifetime id of this ObjectGrid shard.	7.1
Revision	Long	The revision of this ObjectGrid shard.	7.1

Returns:
TabularData representing the known lifetimes and revisions of this shard.
Throws:
OpenDataException
Since:
7.1
See Also:
TabularData

---

getPrimaryShardLinks

TabularData **getPrimaryShardLinks**()

Get the shard's list of foreign or domestic linked primaries.

This method returns a TabularData object representing the current state of each primary shard link.

Each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description
RemoteDomain	String	The catalog service domain name of the remote primary shard..
RemoteContainer	String	The container name of the remote primary shard.
Status	String	The status of the link. Valid states include: online and recovery.

Returns:
TabularData representing the linked primaries
Since:
7.1.1

---

com.ibm.websphere.objectgrid.management

Interface MapMBean

```
public interface MapMBean
```

This MBean interface allows a client process to access different attributes and statistical data about a specific map on a server process. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectMap,name=<map>,partition=<partition id>,objectgrid=<objectgrid>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.0.1, XC10

Method Summary

String	**getContainerName**() Gets the name of the container containing the replication group member for the map associated with this MBean.
double	**getMapBatchUpdateMaxTime**() Gets the maximum batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapBatchUpdateMeanTime**() Gets the mean batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapBatchUpdateMinTime**() Gets the minimum batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapBatchUpdateTotalTime**() Gets the total batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.

long	**getMapCountStatistic**() Gets the map count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getMapGetCountStatistic**() Gets the get count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getMapHitCountStatistic**() Gets the hit count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapHitRateStatistic**() Gets the hit rate attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
String	**getMapName**() Gets the name of the map associated with this MBean.
String	**getMapStatsModule**() Gets a string representation of the `MapStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`
long	**getMapUsedBytes**() Gets the used bytes attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
String	**getObjectGridName**() Gets the name of the ObjectGrid containing the map associated with this MBean.
int	**getPartitionId**() Retrieves the partition identifier for this map instance.
String	**getServerName**() Gets the name of the server containing the replication group member for the map associated with this MBean.
TabularData	**retrieveEntries**(String regex) Operation to iterate through all of the entries in this map, convert the key to string form, then match the string against the regular expression if passed, finally return the matching entries.

com.ibm.websphere.objectgrid.stats.MapStatsModule

retrieveStatsModule()
Gets the MapStatsModule used to retrieve statistics associated with the map for this MBean.

Method Detail

retrieveStatsModule

com.ibm.websphere.objectgrid.stats.MapStatsModule **retrieveStatsModule**()

Gets the MapStatsModule used to retrieve statistics associated with the map for this MBean.

Returns:
A MapStatsModule for statistics associated with this map.

See Also:
MapStatsModule

---

getMapName

String **getMapName**()

Gets the name of the map associated with this MBean.

Returns:
The name of the map.

---

getObjectGridName

String **getObjectGridName**()

Gets the name of the ObjectGrid containing the map associated with this MBean.

Returns:
The name of the ObjectGrid for the map associated with this MBean.

---

getServerName

String **getServerName**()

Gets the name of the server containing the replication group member for the map associated with this MBean.

Returns:
The name of server containing the replication group member for the map associated with this MBean.

---

getMapStatsModule

String **getMapStatsModule**()

Gets a string representation of the `MapStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`

Returns:
The String form of MapStatsModule
See Also:
retrieveStatsModule(), MapStatsModule

---

getMapCountStatistic

long **getMapCountStatistic**()

Gets the map count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The number of entries in the map.
See Also:
retrieveStatsModule(), MapStatsModule.getNumEntries(boolean)

---

getMapHitRateStatistic

double **getMapHitRateStatistic**()

Gets the hit rate attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The hit rate for the map.
See Also:
retrieveStatsModule(), MapStatsModule.getHitRate(boolean)

---

getMapGetCountStatistic

long **getMapGetCountStatistic**()

Gets the get count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The get count for the map.
Since:
7.1
See Also:
retrieveStatsModule(), MapStatsModule.getHitRate(boolean)

---

getMapUsedBytes

long **getMapUsedBytes**()

Gets the used bytes attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Returns:
The number of bytes in use by the map.
Since:
7.1
See Also:
retrieveStatsModule(), MapStatsModule.getUsedBytes(boolean)

---

getMapHitCountStatistic

long **getMapHitCountStatistic**()

Gets the hit count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The hit count for the map.
Since:
7.1
See Also:
retrieveStatsModule(), MapStatsModule.getHitRate(boolean)

---

getMapBatchUpdateMeanTime

double **getMapBatchUpdateMeanTime**()

Gets the mean batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The mean batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getMapBatchUpdateMaxTime

double **getMapBatchUpdateMaxTime**()

Gets the maximum batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The maximum batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getMapBatchUpdateMinTime

double **getMapBatchUpdateMinTime**()

Gets the minimum batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The minimum batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getMapBatchUpdateTotalTime

double **getMapBatchUpdateTotalTime**()

Gets the total batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The total batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getPartitionId

int **getPartitionId**()

Retrieves the partition identifier for this map instance.

Returns:
The partition identifier.
Since:
WAS XD 6.1.0.4

---

getContainerName

String **getContainerName**()

Gets the name of the container containing the replication group member for the map associated with this MBean.

Returns:
The name of container containing the replication group member for the map associated with this MBean.
Since:
8.5

---

retrieveEntries

TabularData **retrieveEntries**(String regex)

Operation to iterate through all of the entries in this map, convert the key to string form, then match the string against the regular expression if passed, finally return the matching entries. This method could potentially return a very large data structure so care should be taken to ensure the regular expression will reduce the number of keys appropriately.

Each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description
KeyName	String	The domain name of this ObjectGrid shard.
LifetimeIndex	Short	The lifetime index for revisioning.
Revision	Long	The revision number of the last update.

Parameters:
regex - the regular expression to apply to the String form of the key. It should be used in narrowing the entries returned. If null, all entries are returned.

Returns:

A table of entries containing the user readable (String) form of the key and some meta information about the entry.

Since:

7.1.1

---

com.ibm.websphere.objectgrid.management

Interface HashIndexMBean

public interface **HashIndexMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific HashIndex on a server process. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=HashIndex,name=<index-name>,partition=<partition id>,objectgrid
=<objectgrid>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1.0.5, XC10

Method Summary

double	**getBatchUpdateCount**() Gets the index's batchupdate count attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindCollisionCount**() Gets the index's collision count attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindCount**() Gets the index's find count attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindDurationTime**() Gets the index's find duration time attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindFailureCount**() Gets the index's failure count attribute loaded up by the `retrieveStatsModule()` method.

double	**getFindResultCount**() Gets the index's result count attribute loaded up by the `retrieveStatsModule()` method.
String	**getParentMapName**() Gets the index's parent map name
com.ibm.websphere.objectgrid.stats.HashIndexStatsModule	**retrieveStatsModule**() Gets the `HashIndexStatsModule` used to retrieve statistics associated with this index

Method Detail

getParentMapName

`String` **getParentMapName**()

Gets the index's parent map name

Returns:
the name of the map which this index belongs to

---

retrieveStatsModule

`com.ibm.websphere.objectgrid.stats.HashIndexStatsModule` **retrieveStatsModule**()

Gets the `HashIndexStatsModule` used to retrieve statistics associated with this index

Returns:
an HashIndexStatsModule for statistics associated with this index
See Also:
`HashIndexStatsModule`

---

getFindCount

`double` **getFindCount**()

Gets the index's find count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the find operation's invocation count for this index
See Also:
`retrieveStatsModule()`, `HashIndexStatsModule.getFindCount(boolean copy)`

---

getFindDurationTime

`double` **getFindDurationTime**()

Gets the index's find duration time attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the find call's duration time for this index in milliseconds
See Also:
`retrieveStatsModule()`, `HashIndexStatsModule.getFindDurationTime(boolean copy)`

---

getFindResultCount

`double` **getFindResultCount**()

Gets the index's result count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the result count for this index and find operation
See Also:
`retrieveStatsModule()`, `HashIndexStatsModule.getFindResultCount(boolean copy)`

---

getFindFailureCount

`double` **getFindFailureCount**()

Gets the index's failure count attribute loaded up by the `retrieveStatsModule()` method.

Returns:

the failure count for this index and find operation

See Also:

retrieveStatsModule(), HashIndexStatsModule.getFindFailureCount(boolean copy)

---

getFindCollisionCount

double **getFindCollisionCount**()

Gets the index's collision count attribute loaded up by the retrieveStatsModule() method.

Returns:

the collision count for this index and find operation

See Also:

retrieveStatsModule(), HashIndexStatsModule.getFindCollisionCount(boolean copy)

---

getBatchUpdateCount

double **getBatchUpdateCount**()

Gets the index's batchupdate count attribute loaded up by the retrieveStatsModule() method.

Returns:

the doBatchUpdate method's invocation count for this index

See Also:

retrieveStatsModule(), MapIndexPlugin.doBatchUpdate(TxID txid, LogSequence sequence), HashIndexStatsModule.getBatchUpdateCount(boolean copy)

---

com.ibm.websphere.objectgrid.management

Interface DynamicServerMBean

All Superinterfaces:

ServerMBean

```
public interface DynamicServerMBean
extends ServerMBean
```

This MBean interface allows a client process to access different attributes about a specific server process in a dynamic environment. The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGridServer,name=<server>,host=<host>,ogServerName=<server
>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

The following notifications are available:

`og.server.container.coregroup.membership.change`

Description:	All core group membership changes detected by the server's core group manager.
UserData:	The number of members in the core group.
Message:	The name of the core group.
Since:	6.1 FIX 3

`com.ibm.websphere.objectgrid.log`

Description:	All log messages detected by the log notification filter. See the `setLogNotificationFilter(String)` attribute.
UserData:	A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The log record severity level.	8.6
LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The originating log record class name.	8.6
SourceMethodName	String	The originating log record method name.	8.6
SequenceNumber	Long	The log record sequence number	8.6
ThreadID	Integer	The originating log record thread id.	8.6
ThrownMessage	Strin	The thrown exception message or empty string, if	8.6

g not available.

Message: The log message.

Since: 8.6

`com.ibm.websphere.objectgrid.ffdc`

Description: All first-failure data captured by the grid server.

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
SourceID	String	The source code generating the exception.	8.6
ProbeID	String	The source code location associated with the exception captured.	8.6
ExceptionName	String	The exception captured.	8.6
Count	Integer	The current occurrence count for the specified exception.	8.6
DateOfFirstOccurrence	String	The exception's first occurrence time stamp.	8.6
Label	String	The label associated with the captured exception occurrence.	8.6

Message: A notification was generated on the server for a new exception.

Since: 8.6

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static String	**COUNT**
static String	**DATE_OF_FIRST_OCCURRENCE**
static String	**EXCEPTION_NAME**

[String](#)	
static [String](#)	**FFDC_INCIDENT_DATA**
static [String](#)	**FFDC_NOTIFICATION**
static [String](#)	**LABEL**
static [String](#)	**LEVEL_NAME**
static [String](#)	**LOG_MESSAGE_NOTIFICATION** The constant identifying the notification type for log messages.
static [String](#)	**LOG_RECORD_DATA**

static String	**LOGGER_NAME**
static String	**PROBE_ID**
static String	**SEQUENCE_NUMBER**
static String	**SERVER_COREGROUP_MEMBERSHIP_CHANGE** A constant identifying the notification type for all core group membership changes detected by this server.
static String	**SOURCE_CLASS_NAME**
static String	**SOURCE_ID**

String	
static String	**SOURCE_METHOD_NAME**
static String	**THREAD_ID**
static String	**THROWN_MESSAGE**

Method Summary

void	**checkFFDCNotification**() Triggers a simulated exception to be captured as a first-failure data capture (FFDC) event (and subsequently broadcasted as a JMX notification), as a means to test and verify the monitoring being enabled on the server.
void	**checkLoggingNotification**() Generates a set of log records with various severity levels, providing a simple means to test and verify when the JMX notification monitoring being enabled on the grid server.
int	**getAvailableProcessors**() Returns the number of available processors for the JVM hosting this server.
CompositeData	**getEnvironmentInfo**() Retrieve the environment information for the server (host name, WebSphere eXtreme Scale version, and additional information).
lo	**getFreeMemory**() Returns the available memory in bytes for the JVM hosting this server.

ng	
String	**getHostName**() Returns the host name for this process.
String	**getLogNotificationFilter**() Retrieves current regular expression filter being applied while screening log record messages before being broadcasted as JMX notifications by the grid server as a JMX Notification of type:
long	**getMaxMemory**() Returns the maximum memory in bytes for the JVM hosting this server.
TabularData	**getPrimaryRevisions**() Provides revisions for all primary shards in the container in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:
TabularData	**getRevisions**() Provides revisions for all shards in the server in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:
boolean	**getSafeToShutdown**() Returns true if a replica exists for each primary hosted on this server.
String	**getStatsSpec**() Retrieve the current statistics specification for the server.
long	**getTotalMemory**() Returns the total memory in bytes for the JVM hosting this server.
String	**getTraceSpec**() Retrieve the current trace specification for the server.

String	**getZoneName**() Returns the zone name for this process
void	**setLogNotificationFilter**(String regExFilter) Sets the regular expression filter to be applied in screening log record messages before being broadcasted by the grid server as a JMX Notification of type:
void	**setStatsSpec**(String statsSpec) Set the statistics specification for the server.
void	**setTraceSpec**(String traceSpec) Set the trace specification for the server.

Methods inherited from interface com.ibm.websphere.objectgrid.management.ServerMBean
getServerName, modifyServerTraceSpec, stopServer

Field Detail

SERVER_COREGROUP_MEMBERSHIP_CHANGE

static final String **SERVER_COREGROUP_MEMBERSHIP_CHANGE**

A constant identifying the notification type for all core group membership changes detected by this server.

Since:
6.1 FIX 3

See Also:
Constant Field Values

LOG_MESSAGE_NOTIFICATION

static final String **LOG_MESSAGE_NOTIFICATION**

The constant identifying the notification type for log messages.

Since:
8.6, XC10 2.5

See Also:
Constant Field Values

FFDC_NOTIFICATION

static final String **FFDC_NOTIFICATION**

See Also:
Constant Field Values

LOG_RECORD_DATA

static final String **LOG_RECORD_DATA**

See Also:
Constant Field Values

THROWN_MESSAGE

static final String **THROWN_MESSAGE**

See Also:
Constant Field Values

THREAD_ID

static final String **THREAD_ID**

See Also:
Constant Field Values

SEQUENCE_NUMBER

static final String **SEQUENCE_NUMBER**

See Also:
Constant Field Values

SOURCE_METHOD_NAME

static final String **SOURCE_METHOD_NAME**

See Also:
Constant Field Values

SOURCE_CLASS_NAME

static final String **SOURCE_CLASS_NAME**

See Also:
Constant Field Values

LOGGER_NAME

static final String **LOGGER_NAME**

See Also:
Constant Field Values

LEVEL_NAME

static final String **LEVEL_NAME**

See Also:
Constant Field Values

FFDC_INCIDENT_DATA

static final String **FFDC_INCIDENT_DATA**

See Also:
Constant Field Values

SOURCE_ID

static final String **SOURCE_ID**

See Also:
Constant Field Values

PROBE_ID

static final String **PROBE_ID**

See Also:
Constant Field Values

LABEL

static final String **LABEL**

See Also:
Constant Field Values

DATE_OF_FIRST_OCCURRENCE

static final String **DATE_OF_FIRST_OCCURRENCE**

See Also:
Constant Field Values

COUNT

static final String **COUNT**

See Also:
Constant Field Values

EXCEPTION_NAME

static final String **EXCEPTION_NAME**

See Also:
Constant Field Values

Method Detail

getAvailableProcessors

int **getAvailableProcessors**()

Returns the number of available processors for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.availableProcessors()]

---

getFreeMemory

`long` **`getFreeMemory`**`()`

Returns the available memory in bytes for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.freeMemory()]

---

getMaxMemory

`long` **`getMaxMemory`**`()`

Returns the maximum memory in bytes for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.maxMemory()]

---

getTotalMemory

`long` **`getTotalMemory`**`()`

Returns the total memory in bytes for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.totalMemory()]

---

getHostName

[String] **`getHostName`**`()`

Returns the host name for this process.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[InetAddress.getHostName()]

---

getZoneName

[String] **`getZoneName`**`()`

Returns the zone name for this process

Returns:

the zone name that was included in the properties used to start the server or DefaultZone if no zone name was used

---

getSafeToShutdown

`boolean` **`getSafeToShutdown`**`()`

Returns true if a replica exists for each primary hosted on this server. Returns false if the server has the only copy of data.

Returns:
If server is safe to shutdown.

---

getStatsSpec

`String` **`getStatsSpec`**`()`

Retrieve the current statistics specification for the server.

Returns:
a string representation of the statistics specification.
Since:
7.1
See Also:
`StatsSpec`

---

setStatsSpec

`void` **`setStatsSpec`**`(String statsSpec)`

Set the statistics specification for the server.

Parameters:
`statsSpec` - the statistics specification string.
Since:
7.1
See Also:
`StatsSpec`

---

getTraceSpec

`String` **`getTraceSpec`**`()`

Retrieve the current trace specification for the server.

Returns:
the trace specification string.
Since:
7.1

---

setTraceSpec

`void` **`setTraceSpec`**`(String traceSpec)`

Set the trace specification for the server.

Parameters:

`traceSpec` - the statistics specification string.

Since:

7.1

See Also:

`ObjectGridManager.setTraceSpecification(String)`

---

getEnvironmentInfo

`CompositeData` **`getEnvironmentInfo`**`()`

Retrieve the environment information for the server (host name, WebSphere eXtreme Scale version, and additional information). The CompositeData contains the following items:

Item Name	Type	Description
JMXServicePort	String	JMX Service Port
WASServerName	String	WebSphere Application Server Full Server Name
JVMVersion	String	JAVA Version
JMXConnectorPort	String	JMX Connector Port
IPAddress	String	IP Address
JavaVM	String	JVM Version
WASInstallRoot	String	WebSphere Application Server Product Directory
OSGIFrameworkVersion	String	OSGi Version
ClientPort	String	Client Port
HostName	String	Host name
Timestamp	String	Time Stamp from Server
WASBaseVersion	String	IBM WebSphere Application Server Version
OSName	String	Operating System
XSInstallRoot	String	WebSphere eXtreme Scale Product Directory
OSArch	String	OS Architecture
PeerPort	String	Peer Port
ServerType	String	Server Type
HAManagerPort	String	HAManager Port
JVMInstallPath	String	JAVA Directory
XC10Model	String	Machine Type and Model
JavaRuntimeInfo	String	JVM Runtime Version
xioContainerTCPNonSecure	String	XIO TCP/IP Port
JMXServiceURL	String	JMX Service Port
listenerPort	String	Listener Port (ORB)
ServerName	String	Server Name
WASXDVersion	String	IBM WebSphere Application Server - XD Version
WASExpressVersion	String	IBM WebSphere Application Server - ND Version
JavaBitMode	String	JAVA Bit Mode
XSVersion	String	WebSphere eXtreme Scale Version
JVMVendor	String	JAVA Vendor
OSVersion	String	Operating System Version
xioContainerTCPSecure	String	XIO TCP/IP SSL Port
PID	String	Process ID
WASNDVersion	String	IBM WebSphere Application Server - ND Version
ORB Version	String	ORB Version

Returns:

CompositeData containing the environment information

Since:
8.5, XC10 2.5

---

setLogNotificationFilter

`void` **`setLogNotificationFilter`**(`String` `regExFilter`)

Sets the regular expression filter to be applied in screening log record messages before being broadcasted by the grid server as a JMX Notification of type:

`com.ibm.websphere.objectgrid.log`

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The logging level.	8.6
LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The class name.	8.6
SourceMethodName	String	The method name.	8.6
SequenceNumber	Long	The sequence number	8.6
ThreadID	Integer	The thread id.	8.6
ThrownMessage	String	The thrown exception message or empty string, if not available.	8.6

Message: The log message.

Since: 8.6

Parameters:

`regExFilter` - The regular expression filter to applied in screening log record messages.

Since:
8.6, XC10 2.5

---

getLogNotificationFilter

`String` **`getLogNotificationFilter`**()

Retrieves current regular expression filter being applied while screening log record messages before being broadcasted as JMX notifications by the grid server as a JMX Notification of type:

`com.ibm.websphere.objectgrid.log`

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The logging level.	8.6

LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The class name.	8.6
SourceMethodName	String	The method name.	8.6
SequenceNumber	Long	The sequence number	8.6
ThreadID	Integer	The thread id.	8.6
ThrownMessage	String	The thrown exception message or empty string, if not available.	8.6

Message: The log message.

Since: 8.6

Returns:
Current regular expression filter being applied in screening log record messages.

Since:
8.6, XC10 2.5

---

checkFFDCNotification

`void` **`checkFFDCNotification`**`()`

Triggers a simulated exception to be captured as a first-failure data capture (FFDC) event (and subsequently broadcasted as a JMX notification), as a means to test and verify the monitoring being enabled on the server. The JMX notification generated and broadcasted has the following type and content:

`com.ibm.websphere.objectgrid.ffdc`

Description: All first-failure data captured by the grid server.

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
SourceID	String	The source code generating the exception.	8.6
ProbeID	String	The source code location associated with the exception captured.	8.6
ExceptionName	String	The exception captured.	8.6
Count	Integer	The current occurrence count for the specified exception.	8.6
DateOfFirstOccurrence	String	The exception's first occurrence time stamp.	8.6
Label	String	The label associated with the captured exception occurrence.	8.6

Message: A simulated first-failure data capture (FFDC) exception notification was generated by the server.

Since: 8.6

Since:
8.6, XC10 2.5

checkLoggingNotification

`void` **`checkLoggingNotification`**`()`

Generates a set of log records with various severity levels, providing a simple means to test and verify when the JMX notification monitoring being enabled on the grid server. Each of the JMX notifications generated and broadcasted has the following type and content:

`com.ibm.websphere.objectgrid.log`

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The logging level.	8.6
LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The class name.	8.6
SourceMethodName	String	The method name.	8.6
SequenceNumber	Long	The sequence number	8.6
ThreadID	Integer	The thread id.	8.6
ThrownMessage	String	A simulated log info/warning/error notification was generated by the server	8.6

Message: A simulated log info/warning/error notification was generated by the server.

Since: 8.6

Since:

8.6, XC10 2.5

---

getRevisions

`TabularData` **`getRevisions`**`()`

Provides revisions for all shards in the server in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Grid	String	The object grid name.	8.6
MapSet	String	The name of the mapSet.	8.6
PartitionNumber	Integer	The number of a given partition.	8.6
ContainerName	String	Name of the container.	8.6
ShardType	String	Type of the shard.	8.6
RevisionState	TabularData	The lifetimeId / revisionNumber info for a given partition.	8.6

The RevisionState TabularData's CompositeData contains the following items:

Item Name	Type	Description	Since
LifetimeId	Long	The LifetimeId for the shard	8.6
Revision	Long	The revision number	8.6

Returns:
A TabularData object with the revision information.
Since:
8.6, XC10 2.5

---

getPrimaryRevisions

`TabularData` **`getPrimaryRevisions`**`()`

Provides revisions for all primary shards in the container in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Grid	String	The object grid name.	8.6
MapSet	String	The name of the mapSet.	8.6
PartitionNumber	Integer	The number of a given partition.	8.6
ContainerName	String	Name of the container.	8.6
ShardType	String	Type of the shard.	8.6
RevisionState	TabularData	The lifetimeId / revisionNumber info for a given partition.	8.6

The RevisionState TabularData's CompositeData contains the following items:

Item Name	Type	Description	Since
LifetimeId	Long	The LifetimeId for the shard	8.6
Revision	Long	The revision number	8.6

Returns:
A TabularData object with the revision information.
Since:
8.6, XC10 2.5

---

com.ibm.websphere.objectgrid.management

Interface ContainerMBean

public interface **ContainerMBean**

This MBean interface allows a client process to perform operations on and get status from an ObjectGrid container running in a dynamic environment. The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGridContainer,name=<server>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
WAS XD 6.1 FIX3, XC10

Field Summary

static String	**INVALID_PARTITION** INVALID_PARTITION indicates that no partition was found for the requested shard.
static String	**MAPSET_UNSUPPORTED_ON_CONTAINER** MAPSET_UNSUPPORTED_ON_CONTAINER indicates that an attempt was made to reserve a shard from a map set that is not supported on this container.
static String	**QUIESCE_COMPLETE** QUIESCE_COMPLETE is the MBean notification type for a completed quiesce.
s	

static String	**RELEASE_SUCCESSFUL** RELEASE_SUCCESSFUL indicates that the attempt to release the shard was successful.
static String	**RELEASE_UNSUPPORTED_WITH_PER_CONTAINER** RELEASE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy.
static String	**RESERVATION_PRIOR_TO_INITIAL_PLACEMENT** RESERVATION_PRIOR_TO_INITIAL_PLACEMENT indicates that the attempt to reserve the shard was processed successfully.
static String	**RESERVATION_SUCCESSFUL** RESERVATION_SUCCESSFUL indicates that the attempt to reserve the shard was successful.
static String	**RESERVE_UNSUPPORTED_WITH_PER_CONTAINER** RESERVE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy.
static String	**SHARD_ALREADY_RESERVED** SHARD_ALREADY_RESERVED indicates that the shard is already reserved elsewhere and cannot be reserved on the specified container.
sta	

static String	**SHARD_NOT_RESERVED_ON_CONTAINER** SHARD_NOT_RESERVED_ON_CONTAINER indicates that the attempt to release the shard from the requesting container failed because the specified shard was not found to be reserved by the requesting container.

Method Summary

int	**getActivatedShardCount**() Retrieve the total number of shards that have been activated for the life of this ObjectGrid container.
int	**getActiveShardCount**() Retrieve the number of active shards hosted in this ObjectGrid container.
String	**getContainerName**() Retrieve the name of the container.
int	**getDeactivatedShardCount**() Retrieve the total number of shards that have been deactivated for the life of this ObjectGrid container.
String	**getDomainName**() Retrieve the name of the catalog server grouping administering this container.
String	**getStatus**() Retrieve the status information for the shards in this container.
String	**getZoneName**() Retrieve the name of the zone grouping that this container belongs to.
int	**quiesceContainer**(Boolean inQuiesce) Prepare the container for a potential shutdown by moving replica shards, verifying that primaries have required sync replicas and preventing the placement of new shards.
String	**release**(String objectGridName, String mapSetName, String partitionName) Release a shard that has been previously reserved by this container.
String	**reserve**(String objectGridName, String mapSetName, String partitionName, String shardType) Reserve a specific shard on this container.

String	**retrieveStatus**(String objectGridName, String mapSetName) Retrieve the status information for the shards in this container, filtered by ObjectGrid and/or mapset.
void	**teardown**() Tears down and stops the container in a way to allow partitions to be moved to new locations.
void	**terminate**() Terminates a container without coordinating partition movement, partitions will failover.

Field Detail

QUIESCE_COMPLETE

static final String **QUIESCE_COMPLETE**

QUIESCE_COMPLETE is the MBean notification type for a completed quiesce.

See Also:
quiesceContainer(Boolean), Constant Field Values

RESERVATION_SUCCESSFUL

static final String **RESERVATION_SUCCESSFUL**

RESERVATION_SUCCESSFUL indicates that the attempt to reserve the shard was successful. The shard is reserved by the requesting container.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

RESERVATION_PRIOR_TO_INITIAL_PLACEMENT

static final String **RESERVATION_PRIOR_TO_INITIAL_PLACEMENT**

RESERVATION_PRIOR_TO_INITIAL_PLACEMENT indicates that the attempt to reserve the shard was processed successfully. However, since initial placement has not yet occurred, the reserved shard is not immediately moved to the requesting container. The shard will be placed on the container when initial placement is triggered.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

SHARD_ALREADY_RESERVED

static final String **SHARD_ALREADY_RESERVED**

SHARD_ALREADY_RESERVED indicates that the shard is already reserved elsewhere

and cannot be reserved on the specified container. The shard must be released from the owning container before it can be reserved again.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

INVALID_PARTITION

static final String **INVALID_PARTITION**

INVALID_PARTITION indicates that no partition was found for the requested shard.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

RESERVE_UNSUPPORTED_WITH_PER_CONTAINER

static final String **RESERVE_UNSUPPORTED_WITH_PER_CONTAINER**

RESERVE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy. Shard reservation is not supported with this placement strategy.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

RELEASE_SUCCESSFUL

static final String **RELEASE_SUCCESSFUL**

RELEASE_SUCCESSFUL indicates that the attempt to release the shard was successful. The shard is no longer reserved by this container. The shard is free to migrate, but it is not forced to migrate.

Since:
7.0.0.0 FIX1
See Also:
release(String, String, String), Constant Field Values

SHARD_NOT_RESERVED_ON_CONTAINER

static final String **SHARD_NOT_RESERVED_ON_CONTAINER**

SHARD_NOT_RESERVED_ON_CONTAINER indicates that the attempt to release the shard from the requesting container failed because the specified shard was not found to be reserved by the requesting container. Only the container owning the reservation may release a shard.

Since:
7.0.0.0 FIX1
See Also:
release(String, String, String), Constant Field Values

RELEASE_UNSUPPORTED_WITH_PER_CONTAINER

`static final String` **`RELEASE_UNSUPPORTED_WITH_PER_CONTAINER`**

RELEASE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy. Shard release is not supported with this placement strategy.

Since:
7.0.0.0 FIX1

See Also:
`release(String, String, String)`, Constant Field Values

MAPSET_UNSUPPORTED_ON_CONTAINER

`static final String` **`MAPSET_UNSUPPORTED_ON_CONTAINER`**

MAPSET_UNSUPPORTED_ON_CONTAINER indicates that an attempt was made to reserve a shard from a map set that is not supported on this container. Only map sets that were included in the deployment policy at container initialization are supported to run on this container.

Since:
7.1

See Also:
`reserve(String, String, String, String)`, Constant Field Values

Method Detail

teardown

`void` **`teardown`**`()`

Tears down and stops the container in a way to allow partitions to be moved to new locations.

---

terminate

`void` **`terminate`**`()`

Terminates a container without coordinating partition movement, partitions will failover.

---

getActiveShardCount

`int` **`getActiveShardCount`**`()`

Retrieve the number of active shards hosted in this ObjectGrid container.

Returns:
The current number of active shards.

---

getActivatedShardCount

`int` **`getActivatedShardCount`**`()`

Retrieve the total number of shards that have been activated for the life of this

ObjectGrid container.

Returns:
The number of activated shards.

---

getDeactivatedShardCount

`int` **`getDeactivatedShardCount`**`()`

Retrieve the total number of shards that have been deactivated for the life of this ObjectGrid container.

Returns:
The number of deactivated shards

---

getDomainName

`String` **`getDomainName`**`()`

Retrieve the name of the catalog server grouping administering this container.

Returns:
The domain name.

---

getZoneName

`String` **`getZoneName`**`()`

Retrieve the name of the zone grouping that this container belongs to.

Returns:
The name of the zone.

---

quiesceContainer

`int` **`quiesceContainer`**`(Boolean inQuiesce)`

Prepare the container for a potential shutdown by moving replica shards, verifying that primaries have required sync replicas and preventing the placement of new shards.

Parameters:
`inQuiesce` - Initiate quiesce mode (true) or cancel quiesce mode (false)
Returns:
The number of replicas moved off of the ObjectGrid container
See Also:
`QUIESCE_COMPLETE`

---

getStatus

`String` **`getStatus`**`()`

Retrieve the status information for the shards in this container.

Returns:
The status information for the shards in this container.

---

retrieveStatus

```
String retrieveStatus(String objectGridName,
                      String mapSetName)
```

Retrieve the status information for the shards in this container, filtered by ObjectGrid and/or mapset. For example, calling retrieveStatus with "og1" and "ms1" as parameters will return the partition status for those partitions in ObjectGrid og1 and mapset ms1. Passing in an empty string ("") objectGridName or mapSetName will return all of the partitions, since the empty string acts as a wildcard. Passing in the empty string for both parameters will return the same status as calling getStatus().

Parameters:

`objectGridName` - The name of the ObjectGrid for which the status is requested.
`mapSetName` - The name of the mapset within the ObjectGrid for which the status is requested.

Returns:

The status information for the shards in this container.

---

reserve

```
String reserve(String objectGridName,
               String mapSetName,
               String partitionName,
               String shardType)
```

Reserve a specific shard on this container. Calling this method will cause the requested shard to move to this container. The shard can be moved to this container only if it is not reserved elsewhere. Calling this method prior to initial placement will pre-reserve the shard so that it will be placed onto this container when initial placement occurs. If non-reserved shard for the same partition is on this container prior to reservation, the non-reserved shard will be moved off the container upon reservation. A reserved shard will not be moved off of this container until it is released or the container is stopped.

Parameters:

`objectGridName` - the ObjectGrid containing the shard
`mapSetName` - the map set containing the shard
`partitionName` - the partition containing the shard
`shardType` - the type of shard. Currently, only primary shards can be reserved: `ShardMBean.TYPE_PRIMARY`

Returns:

the return code indicating the result of the reserve request

Throws:

`IllegalArgumentException` - if any of the arguments are null or the empty String. Also thrown if shardType is not `ShardMBean.TYPE_PRIMARY`

Since:

7.0.0.0 FIX1

See Also:

`ShardMBean.TYPE_PRIMARY`, `release(String, String, String)`, `RESERVATION_SUCCESSFUL`, `RESERVATION_PRIOR_TO_INITIAL_PLACEMENT`, `SHARD_ALREADY_RESERVED`, `INVALID_PARTITION`, `RESERVE_UNSUPPORTED_WITH_PER_CONTAINER`

---

release

```
String release(String objectGridName,
               String mapSetName,
               String partitionName)
```

Release a shard that has been previously reserved by this container. This container can only release shards that it has reserved. Releasing the shard does not guarantee the

shard will be moved. The shard may remain on this container. However, it will not be explicitly bound to this container. Releasing a shard allows the shard to move freely to other containers or to be reserved by another container.

Parameters:

`objectGridName` - the ObjectGrid containing the shard
`mapSetName` - the map set containing the shard
`partitionName` - the partition containing the shard

Returns:

the return code indicating the result of the release request

Throws:

`IllegalArgumentException` - if any of the arguments are null or the empty String

Since:

7.0.0.0 FIX1

See Also:

`reserve(String, String, String, String)`, `RELEASE_SUCCESSFUL`, `SHARD_NOT_RESERVED_ON_CONTAINER`, `RELEASE_UNSUPPORTED_WITH_PER_CONTAINER`

---

getContainerName

`String` **`getContainerName`**`()`

Retrieve the name of the container. The container name is based on the server name and includes a suffix which uniquely identifies the container within the server.

Returns:

the name of the container

Since:

7.1

---

com.ibm.websphere.objectgrid.management

Interface CatalogServiceManagementMBean

public interface **CatalogServiceManagementMBean**

This MBean interface allows user to manipulate the behaviors of heartbeat and leader manager and other catalog service specific actions. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=CatalogService
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
7.1, XC10

Field Summary

static String	**COREGROUP_MEMBERSHIP_CHANGE** Constant representing a core group membership change notification.
static int	**HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE** Constant representing a heartbeat frequency level at an aggressive rate.
static int	**HEARTBEAT_FREQUENCY_LEVEL_RELAXED** Constant representing a heartbeat frequency level at relaxed rate.
static int	**HEARTBEAT_FREQUENCY_LEVEL_TYPICAL** Constant representing a heartbeat frequency level at a typical rate.

int	
static String	**MANAGEMENT_CONCENTRATOR_STATUS_STARTED**
static String	**MANAGEMENT_CONCENTRATOR_STATUS_STOPPED**
static String	**ORB** Constant representing the ORB or Object Request Broker transport
static String	**SERVER_EVENT_STARTED** Constant representing an eXtreme Scale server start notification.
static String	**SERVER_EVENT_STOPPED** Constant representing an eXtreme Scale server stop notification.
static String	**XIO** Constant representing the eXtremeIO transport

[n g](#)	

Method Summary

[T a b u l a r D a t a](#)	**getContainerReplicationState**() Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross containers within one domain.
[T a b u l a r D a t a](#)	**getDomainReplicationState**() Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross domains.
int	**getHeartBeatFrequencyLevel**() Retrieves the heartbeat frequency level.
[S t r i n g](#)	**getManagementConcentratorStatus**() Returns the String representation of the management concentrator status.
int	**getNumberOfServers**() Retrieves the number eXtreme Scale servers that are currently registered with the catalog service.
[C o m p o s i t e D a t a](#)	**getServers**() Retrieves a CompositeData of each eXtreme Scale server that is currently registered with the catalog service.
[S t r i n g](#)	**getTransport**() Returns the transport used by the catalog service domain.
boolean	**isPrimary**() Provides indication if the catalog is primary.

void	**logMessage**(String level, String message) Provides support for logging user messages from processes outside the catalog and/or container servers.
void	**startManagementConcentrator**() Starts the Management Concentrator.
void	**stopManagementConcentrator**() Stops the Management Concentrator.

Field Detail

MANAGEMENT_CONCENTRATOR_STATUS_STARTED

static final String **MANAGEMENT_CONCENTRATOR_STATUS_STARTED**

See Also:
Constant Field Values

MANAGEMENT_CONCENTRATOR_STATUS_STOPPED

static final String **MANAGEMENT_CONCENTRATOR_STATUS_STOPPED**

See Also:
Constant Field Values

COREGROUP_MEMBERSHIP_CHANGE

static final String **COREGROUP_MEMBERSHIP_CHANGE**

Constant representing a core group membership change notification. The user data associated with this notification is a CompositeData.

The CompositeData includes the following items:

Item Name	Type	Description
MemberName	String	The name of the server that is included in the core group.

See Also:
Constant Field Values

SERVER_EVENT_STARTED

static final String **SERVER_EVENT_STARTED**

Constant representing an eXtreme Scale server start notification.

The UserData argument of the Notification includes a TabularData that includes information for each of the servers. Each CompositeData (row in the TabularData) contains the following items:

	Description

Item Name	Type	
HAPort	String	The port number of the high availability manager.
Host	String	The host/ip address of the server.
JMXServiceURL	String	The JMX service url used to access the server.
ServerName	String	The name of the server.
ZoneName	String	The name of the zone that the server belongs.

See Also:
Constant Field Values

SERVER_EVENT_STOPPED

`static final String` **`SERVER_EVENT_STOPPED`**

Constant representing an eXtreme Scale server stop notification.

The UserData argument of the Notification includes a TabularData instance where each CompositeData contains the following items:

Item Name	Type	Description
ServerName	String	The name of the server.

See Also:
Constant Field Values

HEARTBEAT_FREQUENCY_LEVEL_TYPICAL

`static final int` **`HEARTBEAT_FREQUENCY_LEVEL_TYPICAL`**

Constant representing a heartbeat frequency level at a typical rate.

A typical heartbeat frequency allows reasonable failover detection and resource utilization. This value is the default.

See Also:
Constant Field Values

HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE

`static final int` **`HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE`**

Constant representing a heartbeat frequency level at an aggressive rate.

An increased heartbeat frequency allows failures to be detected more quickly, but can also uses additional CPU and network resources. This level is more sensitive to missing heartbeats when the server is stressed.

See Also:
Constant Field Values

HEARTBEAT_FREQUENCY_LEVEL_RELAXED

`static final int` **`HEARTBEAT_FREQUENCY_LEVEL_RELAXED`**

Constant representing a heartbeat frequency level at relaxed rate.

A decreased heartbeat frequency increases the time to detect failures, but also decreases

CPU and network utilization.

See Also:
Constant Field Values

ORB

```
static final String ORB
```

Constant representing the ORB or Object Request Broker transport

See Also:
Constant Field Values

XIO

```
static final String XIO
```

Constant representing the eXtremeIO transport

See Also:
Constant Field Values

Method Detail

getHeartBeatFrequencyLevel

```
int getHeartBeatFrequencyLevel()
```

Retrieves the heartbeat frequency level.

Valid values include:

- HEARTBEAT_FREQUENCY_LEVEL_TYPICAL
- HEARTBEAT_FREQUENCY_LEVEL_RELAXED
- HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE

Returns:
the heartbeat frequency level: -1, 0 or 1 as defined by the constants that begin with name HEARTBEAT_FREQUENCY_LEVEL.

---

getServers

```
CompositeData getServers()
```

Retrieves a CompositeData of each eXtreme Scale server that is currently registered with the catalog service.

The CompositeData includes the following items:

Item Name	Type	Description
serverName	String	The name of the server that is registered with the catalog service.

Returns:
the CompositeData representing the currently registered eXtreme Scale servers.

---

getNumberOfServers

`int` **`getNumberOfServers`**`()`

Retrieves the number eXtreme Scale servers that are currently registered with the catalog service.

Returns:
the number of registered eXtreme Scale servers.

---

logMessage

`void` **`logMessage`**`(String level,`
`String message)`

Provides support for logging user messages from processes outside the catalog and/or container servers. Example: XC10 surfaced SNMP trap messages can be flowed from the SNMP agent which throws traps in the console server (sMash) process, not typically an XS catalog/container server.

Parameters:
`level` - name describing the severity of the event which is compatible with java.util.logging.Level.parse(String name) where name may be either level name (ex. "SEVERE") or an integer value (ex. "1000") - @see `Level.parse(String)`
`message` - for the end user (already sNLS rendered)

Since:
8.6, XC10 2.5

---

isPrimary

`boolean` **`isPrimary`**`()`

Provides indication if the catalog is primary.

Returns:
true for primary catalog.

Since:
8.6, XC10 2.5

---

getContainerReplicationState

`TabularData` **`getContainerReplicationState`**`()`

Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross containers within one domain. For a given container, outstanding out-bound revisions need to be replicated from primary shards located in this container into replicas located in other containers. In similar way, outstanding in-bound revisions need to be replicated from primary shards located in other containers into corresponding replicas located in this container. This operation can be used to check the differences in data revisions between containers within one domain.

The result is a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Container	String	The container ID.	8.6
OutboundRevisions	Long	Number of out-bound revisions for container.	8.6
InboundRevisions	Long	Number of in-bound revisions for container.	8.6

Returns:

A TabularData object with the container replication state information.

Since:

8.6, XC10 2.5

---

getDomainReplicationState

`TabularData` **`getDomainReplicationState`**`()`

Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross domains. Outstanding out-bound revisions need to be replicated from local primary shards into corresponding remote primary shards. Outstanding in-bound revisions need to be replicated from remote primary shards into corresponding local primary shards. This operation can be used to check the differences in data revisions between different domains linked by MMR replication.

The result is a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Domain	String	The domain name.	8.6
Container	String	The container ID.	8.6
OutboundRevisions	Long	Number of out-bound revisions for container.	8.6
InboundRevisions	Long	Number of in-bound revisions for container.	8.6

Returns:

A TabularData object with the domain replication state information.

Since:

8.6, XC10 2.5

---

getTransport

`String` **`getTransport`**`()`

Returns the transport used by the catalog service domain.

Returns:

String containing the transport type

Since:

8.6, XC10 2.5

See Also:

`ORB`, `XIO`

---

startManagementConcentrator

`void` **`startManagementConcentrator`**`()`

Starts the Management Concentrator. The catalog server will now start listening for log messages.

Since:

8.6.0.2, XC10 2.5

---

stopManagementConcentrator

`void` **`stopManagementConcentrator`**`()`

Stops the Management Concentrator. The catalog server will no longer listen for log

messages. Stopping the Management Concentrator will also stop the Message Center in the web monitoring console.

Since:
8.6.0.2, XC10 2.5

---

getManagementConcentratorStatus

`String` **`getManagementConcentratorStatus()`**

Returns the String representation of the management concentrator status. Status will be either CatalogServiceManagementMBean.MANAGEMENT_CONCENTRATOR_STATUS_STARTED or CatalogServiceManagementMBeanMANAGEMENT_CONCENTRATOR_STATUS_STOPPED.

Returns:
String containing the status

Since:
8.6.0.2, XC10 2.5

---

com.ibm.websphere.objectgrid.management

Interface AgentManagerMBean

public interface **AgentManagerMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific Agent on a server process. The Agent Manager MBean is scoped at the map level and therefore can access statistical data for every agent run against the specified map. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=AgentManager,name=Agent-<map>,partition=<partition id>,objectgr
id=<objectgrid>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1.0.5, XC10

Method Summary

double	**getAgentInflationTime**(String agentClassName) Gets the specified agent's inflation time attribute loaded up by the `retrieveStatsModule()` method.
double	**getAgentSerializationTime**(String agentClassName) Gets the specified agent's serialization time attribute loaded up by the `retrieveStatsModule()` method.
double	**getFailureCount**(String agentClassName) Gets the specified agent's failure count attribute loaded up by the `retrieveStatsModule()` method.
double	**getInvocationCount**(String agentClassName) Gets the specified agent's invocation count attribute loaded up by the `retrieveStatsModule()` method.
double	**getPartitionCount**(String agentClassName) Gets the specified agent's partition count attribute loaded up by the `retrieveStatsModule()` method.

double	**getReduceTime**(String agentClassName) Gets the specified agent's total reduce time attribute loaded up by the `retrieveStatsModule()` method.
double	**getResultInflationTime**(String agentClassName) Gets the specified agent's result inflation time attribute loaded up by the `retrieveStatsModule()` method.
double	**getResultSerializationTime**(String agentClassName) Gets the specified agent's result serialization time attribute loaded up by the `retrieveStatsModule()` method.
double	**getTotalDurationTime**(String agentClassName) Gets the specified agent's total run time attribute loaded up by the `retrieveStatsModule()` method.
com.ibm.websphere.objectgrid.stats.AgentStatsMo	**retrieveStatsModule**(String agentClassName) Gets the `AgentStatsModule` used to retrieve statistics associated with the specified agent class

d u l e	

Method Detail

getReduceTime

`double` **`getReduceTime`**(`String` `agentClassName`)

Gets the specified agent's total reduce time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the reduce time for this agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getReduceTime(boolean copy)`

---

getTotalDurationTime

`double` **`getTotalDurationTime`**(`String` `agentClassName`)

Gets the specified agent's total run time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the total run time for this agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getTotalDurationTime(boolean copy)`

---

getAgentSerializationTime

`double` **`getAgentSerializationTime`**(`String` `agentClassName`)

Gets the specified agent's serialization time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to serialize the agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getAgentSerializationTime(boolean copy)`

---

getAgentInflationTime

`double` **`getAgentInflationTime`**(`String` `agentClassName`)

Gets the specified agent's inflation time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to inflate the agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getAgentInflationTime(boolean copy)`

---

getResultInflationTime

`double` **`getResultInflationTime`**`(String agentClassName)`

Gets the specified agent's result inflation time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to inflate the agent results for a given partition in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getResultInflationTime(boolean copy)`

---

getResultSerializationTime

`double` **`getResultSerializationTime`**`(String agentClassName)`

Gets the specified agent's result serialization time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to serialize the agent results for a given partition in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getResultSerializationTime(boolean copy)`

---

getPartitionCount

`double` **`getPartitionCount`**`(String agentClassName)`

Gets the specified agent's partition count attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the number of partitions this agent is sent to

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getPartitionCount(boolean copy)`

---

getFailureCount

`double` **`getFailureCount`**`(String agentClassName)`

Gets the specified agent's failure count attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the failure count for the specified agent

See Also:

retrieveStatsModule(String), AgentStatsModule.getFailureCount(boolean copy)

---

getInvocationCount

double **getInvocationCount**(String agentClassName)

Gets the specified agent's invocation count attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the invocation count for the specified agent

See Also:

retrieveStatsModule(String), AgentStatsModule.getInvocationCount(boolean copy)

---

retrieveStatsModule

com.ibm.websphere.objectgrid.stats.AgentStatsModule **retrieveStatsModule**(String agentClassName)

Gets the `AgentStatsModule` used to retrieve statistics associated with the specified agent class

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

an AgentStatsModule for statistics associated with the specified agent class

See Also:

`AgentStatsModule`

Package com.ibm.websphere.objectgrid.plugins

These are the interfaces for adding plugins to the Grid core framework.

See:

Description

Interface Summary

BeanFactory	Implement this interface to allow bean factories like Spring or Google guice to be integrated.
CacheEntry	This interface represents a cache entry in an ObjectGrid map.
EvictionEventCallback	An instance of `EvictionEventCallback` is passed into the `Evictor` at initialization time.
Evictor	Data contained in a `BackingMap` are evicted when the map is full.
EvictorData	This interface is optionally used by an implementator of the `Evictor` interface.
LogElement	LogElements are the individual entries within a LogSequence.
LogSequence	LogSequence is the ordered list of changes performed against a given map for a given transaction.
LogSequenceFilter	This interface can be used to filter a LogSequence.
MapEventListener	This callback interface is implemented by the application when it wants to receive events about a Map such as the eviction of a map entry.
ObjectGridEventGroup	This is a set of single method interfaces for fine grained events delivered for an ObjectGrid.
ObjectGridEventGroup.ShardEvents	These events are fired when a shard is made a primary shard and when the shard is demoted from a primary.
ObjectGridEventGroup.ShardLifecycle	These events are fired when an ObjectGrid shard is initialized and destroyed.
ObjectGridEventGroup.TransactionEvents	These events are called every single transaction.
ObjectGridEventListener	This interface is used to create an implementation of an event listener for an ObjectGrid.
ReplicationMapListener	**Deprecated.** *The client replicated map function is deprecated in version 8.6.*
Transaction	Calling methods on a `Session` will send corresponding events to the

Callback	TransactionCallback.
TransactionCallback.BeforeCommit	The BeforeCommit optional mix-in interface for the TransactionCallback plug-in interface allows plug-ins to be notified at the beginning of a `Session.commit()`.
TransactionCallback.BeforeCommit.TransactionContext	The TransactionContext identifies various information that's available to the beforeCommit() method.
ValueProxyInfo	This interface can be used by value objects to inform a BackingMap or Loader which attribute(s) have been dirtied.

Class Summary

EvictorData.SpecialEvictorData	Special value class used for representing the key not being found in the `BackingMap`.
LogElement.Type	The `Type` class is used to represent a `LogElement` type.

Exception Summary

CacheEntryException	This exception indicates an error occurred during a cache entry operation.
LoaderException	This exception is the base exception for any exceptions encountered by a `Loader`.
TransactionCallbackException	This exception is thrown when a method call to TransactionCallback fails.

Package com.ibm.websphere.objectgrid.plugins Description

These are the interfaces for adding plugins to the Grid core framework.

Overview

These plugins can be added into ObjectGrid in several ways such as xml configuration, programmatically adding, or using annotation.

Annotation based callbacks

ObjectGrid when running on Java 5 will start to use an annotated method callback system. This means that objects can be registered as callbacks or listeners. The methods on the object must be annotated as to be invoked for a certain event. Unannotated methods are not invoked. The name of the method is unimportant. The method arguments and return type must be the same as expected for the callback method.

Why?

Usually, callbacks are specified using an interface. This works well but results in a possible performance loss as all methods on the interface will be invoked by the ObjectGrid even

though the application is only interested in a single event. This wastes precious resources. Another issue is when we need to add a new event. Adding a new method to an existing interface breaks back wards compatibility. We can make a new interface extending the old one with the new methods but this is also undesirable as soon there are many interfaces in the hierarchy as new events are added. The annotation system allows the application to only mark methods to be called avoiding the first problem and if new event types are added they have no impact on existing callback objects. Newer applications can add a method and annotate it with the new event annotation to receive the event.

---

com.ibm.websphere.objectgrid.plugins

Interface ValueProxyInfo

public interface **ValueProxyInfo**

This interface can be used by value objects to inform a BackingMap or Loader which attribute(s) have been dirtied. This mechanism allows the BackingMap and Loader to interrogate the set of changed attributes in the value object instead of just assuming the whole value object has been updated. For this to be useful, the application must only use the getter and setter methods defined for the value object's interface.

Since:

WAS XD 6.0, XC10

See Also:

CopyMode.COPY_ON_WRITE, BackingMap.setCopyMode(CopyMode, Class), Loader.batchUpdate(TxID, LogSequence)

Method Summary

void	**ibmClearDirtyAttributes**() Clears the list of dirty attributes.
List	**ibmGetDirtyAttributes**() Returns a list of dirty attributes based on the value interface set on the map.
Object	**ibmGetRealValue**() Returns the real value object this proxy represents.

Method Detail

ibmGetDirtyAttributes

List **ibmGetDirtyAttributes**()

Returns a list of dirty attributes based on the value interface set on the map.

The attribute name is always starts with an upper case letter. For example, if the setter for the attribute is setPrice then 'Price' is the string returned here. The runtime uses `substring(3)` of the setter method name as the attribute name.

Returns:

List of attribute names (Strings)

ibmGetRealValue

`Object` **`ibmGetRealValue`**`()`

Returns the real value object this proxy represents.

Needed internally by the BackingMap to return a separate proxy for each transaction.

Returns:
actual value object.

---

ibmClearDirtyAttributes

`void` **`ibmClearDirtyAttributes`**`()`

Clears the list of dirty attributes.

---

com.ibm.websphere.objectgrid.plugins

Class TransactionCallbackException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.TransactionCallbackException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException, ReplicationVotedToRollbackTransactionException

---

```
public class TransactionCallbackException
extends ObjectGridException
```

This exception is thrown when a method call to TransactionCallback fails.

Since:

WAS XD 6.0, XC10

See Also:

TransactionCallback, Serialized Form

Constructor Summary

TransactionCallbackException() Constructs a new TransactionCallbackException with `null` as its detail message.
TransactionCallbackException(String message) Constructs a new TransactionCallbackException with the specified detail message.
TransactionCallbackException(String message, Throwable cause) Constructs a new TransactionCallbackException with the specified detail message and cause.
TransactionCallbackException(Throwable cause) Constructs a new TransactionCallbackException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionCallbackException

`public` **`TransactionCallbackException`**`()`

Constructs a new TransactionCallbackException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

TransactionCallbackException

`public` **`TransactionCallbackException`**`(String message)`

Constructs a new TransactionCallbackException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

TransactionCallbackException

`public` **`TransactionCallbackException`**`(String message,`
`Throwable cause)`

Constructs a new TransactionCallbackException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionCallbackException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

TransactionCallbackException

`public` **`TransactionCallbackException`**`(Throwable cause)`

Constructs a new TransactionCallbackException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains

the class and detail message of `cause`). This constructor is useful for TransactionCallbackExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid.plugins

Interface TransactionCallback

All Known Subinterfaces:

TransactionCallback.BeforeCommit

All Known Implementing Classes:

WebSphereTransactionCallback

`public interface` **`TransactionCallback`**

Calling methods on a `Session` will send corresponding events to the TransactionCallback. An ObjectGrid can have zero or one TransactionCallback. BackingMaps defined on an ObjectGrid with a TransactionCallback should have corresponding Loaders.

A TransactionCallback works with `Loader`s and place transaction specific objects in slots on the `TxID` object that `Loader`s can obtain. Examples are database connections, prepared statement caches, etc. The TransactionCallback should reserve slots in the `TxID` by calling `ObjectGrid.reserveSlot(String)` using the name `TxID.SLOT_NAME`. The TransactionCallback can then put an object at that index in the `TxID`. A `Loader` can retrieve the index used by the `TransactionCallback` by calling an internal method on the `TransactionCallback`'s implementation. A reference to the configured `TransactionCallback` can be found using the `TxID.getSession().getObjectGrid().getTransactionCallback()` code sequence.

A `TransactionCallback` implementation that also implements the `ObjectGridLifecycleListener` interface will be automatically added as an `EventListener` on the `ObjectGrid` when the callback is set on the object grid.

A TransactionCallback may implement the `ObjectGridPlugin` interface in order to receive enhanced ObjectGrid plug-in lifecycle method calls. The plug-in is also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Since:

WAS XD 6.0, XC10

See Also:

`Loader`, `ObjectGrid.addEventListener(EventListener)`, `ObjectGrid.getTransactionCallback()`, `ObjectGrid.reserveSlot(String)`, `ObjectGrid.setTransactionCallback(TransactionCallback)`, `Session.getObjectGrid()`, `TxID.putSlot(int, Object)`, `TxID.getSlot(int)`, `TxID.getSession()`

Nested Class Summary

static interface	**TransactionCallback.BeforeCommit** The BeforeCommit optional mix-in interface for the TransactionCallback plug-in interface allows plug-ins to be notified at the beginning of a `Session.commit()`.

face

Method Summary

void	**begin**(TxID id) Invoked when starting a Session transaction.
void	**commit**(TxID id) Invoked when committing a Session transaction.
void	**initialize**(ObjectGrid objectGrid) Invoked when an `ObjectGrid` is initialized.
boolean	**isExternalTransactionActive**(Session session) Called when an application attempts to use a Session with no transaction active.
void	**rollback**(TxID id) Invoked when rolling back a Session transaction.

Method Detail

initialize

```
void initialize(ObjectGrid objectGrid)
                throws TransactionCallbackException
```

Invoked when an `ObjectGrid` is initialized.

This method is called so this object can do any implementation specific intialization.

Parameters:

`objectGrid` - A reference to the ObjectGrid.

Throws:

TransactionCallbackException - if an error occurs during processing

See Also:

ObjectGrid.reserveSlot(String)

---

begin

```
void begin(TxID id)
           throws TransactionCallbackException
```

Invoked when starting a Session transaction.

A TransactionCallback can communicate the begin processing (along with the TxID) to the appropriate BackingMap and/or Loader. The Loader may use this signal to start a

corresponding transaction on the underlying connection to a database.

Parameters:
`id` - transaction identifer (TxID)
Throws:
`TransactionCallbackException` - if an error occurs during processing
See Also:
`Session.begin()`, `Session.beginNoWriteThrough()`, `TxID`

---

commit

```
void commit(TxID id)
            throws TransactionCallbackException
```

Invoked when committing a Session transaction.

This method should be used to commit any underlying transaction and return any underlying connection back to the pool. The TxID is provided to determine which transaction is being committed

Parameters:
`id` - transaction identifier (TxID)
Throws:
`TransactionCallbackException` - if an error occurs during processing
See Also:
`begin(TxID)`, `Session.commit()`, `TxID`

---

rollback

```
void rollback(TxID id)
              throws TransactionCallbackException
```

Invoked when rolling back a Session transaction.

This method should be used to roll back any underlying transaction and return any underlying connection back to the pool. The TxID is provided to determine which transaction is being committed

Parameters:
`id` - transaction identifier (TxID)
Throws:
`TransactionCallbackException` - if an error occurs during processing
See Also:
`begin(TxID)`, `Session.rollback()`, `TxID`

---

isExternalTransactionActive

```
boolean isExternalTransactionActive(Session session)
```

Called when an application attempts to use a Session with no transaction active.

The callback could return `true` in which case an auto `Session.begin()` is executed. If `false` is returned, an application exception is thrown indicating no transaction is active. This event is usually used when integrating with a J2EE environment such as WebSphere Application Server.

Parameters:
`session` - the session which the application is using
Returns:

`true` if an auto begin should be done, `false` if this is not the case

See Also:

Session

---

com.ibm.websphere.objectgrid.plugins

Interface TransactionCallback.BeforeCommit

All Superinterfaces:
TransactionCallback

Enclosing interface:
TransactionCallback

```
public static interface TransactionCallback.BeforeCommit
extends TransactionCallback
```

The BeforeCommit optional mix-in interface for the TransactionCallback plug-in interface allows plug-ins to be notified at the beginning of a `Session.commit()`. Implementations can use the `beforeCommit()` method to validate changed data in the transaction and modify the data.

Since:
7.1.1

Nested Class Summary

static interface	**TransactionCallback.BeforeCommit.TransactionContext** The TransactionContext identifies various information that's available to the beforeCommit() method.

Nested classes/interfaces inherited from interface com.ibm.websphere.objectgrid.plugins.TransactionCallback

TransactionCallback.BeforeCommit

Method Summary

void	**beforeCommit**(TransactionCallback.BeforeCommit.TransactionContext ctx) Invoked at the beginning of a `Session.commit()`.

Methods inherited from interface com.ibm.websphere.objectgrid.plugins.TransactionCallback

begin, commit, initialize, isExternalTransactionActive, rollback

Method Detail

beforeCommit

```
void beforeCommit(TransactionCallback.BeforeCommit.TransactionContext ctx)
                  throws TransactionCallbackException
```

Invoked at the beginning of a `Session.commit()`.

Use the `TransactionContext.getLogSequences()` method to retrieve the changes made by this transaction. Use the `TransactionContext.getTxId().getSession()` methods to access the Session. The Session can be used to access ObjectMaps and modify data in the current transaction.

Parameters:

`ctx` - the context of the transaction.

Throws:

`TransactionCallbackException` - if an error occurs during processing. Any exception will roll back the transaction and will be included in the TransactionException thrown to the caller.

See Also:

`Session.commit()`, `TxID`

---

com.ibm.websphere.objectgrid.plugins

Interface TransactionCallback.BeforeCommit.TransactionContext

Enclosing interface:
TransactionCallback.BeforeCommit

```
public static interface TransactionCallback.BeforeCommit.TransactionContext
```

The TransactionContext identifies various information that's available to the beforeCommit() method.

Since:
7.1.1

Method Summary

Collection<LogSequence>	**getLogSequences**() The LogSequences that reflect the pending changes to the map.
TxID	**getTxID**() Retrieve the TxID for the transaction.

Method Detail

getTxID

TxID **getTxID**()

Retrieve the TxID for the transaction.

Returns:
the TxID for the transaction.

---

getLogSequences

`Collection<LogSequence> getLogSequences()`

The LogSequences that reflect the pending changes to the map.

Returns:
the LogSequences.

---

com.ibm.websphere.objectgrid.plugins

Interface ReplicationMapListener

Deprecated. *The client replicated map function is deprecated in version 8.6. Use the ContinuousQueryManager function.*

public interface **ReplicationMapListener**

This interface is used to create an implementation of an event listener for client-side maps that are in replication mode. Registered listeners receive notification callbacks for replication start and exit events and data changes.

Listener instances can be registered with a map using the ClientReplicableMap.enableClientReplication(com.ibm.websphere.objectgrid.ClientReplicableMap.Mode, int[], ReplicationMapListener) method.

Since:
WAS XD 6.1, XC10

Method Summary

void	**onData**(LogSequence logSequence) **Deprecated.** This method is invoked when a data change is replicated to the client replication map.
void	**replicationExits**() **Deprecated.** This method is invoked when replication has been disabled for this map.
void	**replicationStarts**() **Deprecated.** This method is invoked when a snapshot mode replica has been synchronized with the client or a continuous replica has started replicating.

Method Detail

replicationStarts

void **replicationStarts**()

Deprecated.
This method is invoked when a snapshot mode replica has been synchronized with the client or a continuous replica has started replicating.

See also:
ClientReplicableMap.enableClientReplication(com.ibm.websphere.objectgrid.ClientReplicableMap.Mode, int[], ReplicationMapListener)

onData

`void` **`onData`**(`LogSequence` `logSequence`)

Deprecated.
This method is invoked when a data change is replicated to the client replication map.

Parameters:

`logSequence` - the log sequence containing all of the data changes.

---

replicationExits

`void` **`replicationExits`**()

Deprecated.
This method is invoked when replication has been disabled for this map.

See Also:

`ClientReplicableMap.disableClientReplication()`

---

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventListener

All Superinterfaces:

com.ibm.websphere.objectgrid.plugins.EventListener

```
public interface ObjectGridEventListener
extends com.ibm.websphere.objectgrid.plugins.EventListener
```

This interface is used to create an implementation of an event listener for an ObjectGrid. Instances of `ObjectGridEventListener`s are set on the `ObjectGrid` interface. Any significant events are communicated to the application using the methods outlined below. When using Java 5, this callback also supports new callback annotation mechanism.

Since:

WAS XD 6.0, XC10

See Also:

`ObjectGrid.addEventListener(EventListener)`, `ObjectGrid.removeEventListener(EventListener)`, `EventListener`

Method Summary

void	**destroy**() Called when the `ObjectGrid` associated with this listener is destroyed.
void	**initialize**(Session session) Invoked when an `ObjectGrid` is initialized.
void	**transactionBegin**(String txid, boolean isWriteThroughEnabled) Signals the beginning of a `Session` transaction.
void	**transactionEnd**(String txid, boolean isWriteThroughEnabled, boolean committed, Collection changes) Signals the ending of a `Session` transaction.

Method Detail

initialize

```
void initialize(Session session)
```

Invoked when an `ObjectGrid` is initialized.

A usable `Session` instance is passed to this listener to provide all of the necessary access

to the various ObjectGrid objects.

Parameters:

`session` - a Session instance that this listener is associated with.

See Also:

`ObjectGrid.initialize()`

---

transactionBegin

```
void transactionBegin(String txid,
                      boolean isWriteThroughEnabled)
```

Signals the beginning of a `Session` transaction.

A stringified version of the TxID is provided for correlating with the end of the transaction, if so desired. The type of transaction is also provided by the isWriteThroughEnabled boolean parameter.

Parameters:

`txid` - Stringified version of the TxID

`isWriteThroughEnabled` - boolean flag indicating whether the Session transaction was started using the `Session.beginNoWriteThrough()`. method. `false` is passed if `beginNoWriteThrough()` was used.

See Also:

`Session.begin()`, `Session.beginNoWriteThrough()`

---

transactionEnd

```
void transactionEnd(String txid,
                    boolean isWriteThroughEnabled,
                    boolean committed,
                    Collection changes)
```

Signals the ending of a `Session` transaction.

A string version of the TxID is provided for correlating with the begin of the transaction, if so desired. Map changes are also reported with the collection of `LogSequence`s passed to this method. Typical uses of this event are for customers doing custom peer invalidation or peer commit push. This event listener gives them the changes. Calls to this method are made after commit and are sequenced so that they are delivered one by one, not in parallel. The event order is the commit and rollback order.

For an ObjectGridEventListener receiving changes in an `ObjectMap` that is configured to use a `OutputFormat.RAW` for the keys or values, the keys and values objects in the LogSequences will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the implementation class.

Parameters:

`txid` - string version of the TxID

`isWriteThroughEnabled` - boolean flag indicating whether the Session transaction was started using the `Session.beginNoWriteThrough()`. method. `false` is passed if `beginNoWriteThrough()` was used.

`committed` - a boolean flag indicating whether the transaction was committed (`true`) or rolled back (`false`)

`changes` - a Collection of LogSequences representing the changes that were committed or rolled back.

See Also:

LogSequence.isRollback(), Session.begin(), Session.beginNoWriteThrough(), Session.commit(), Session.rollback()

---

destroy

```
void destroy()
```

Called when the ObjectGrid associated with this listener is destroyed.

This method is the opposite of the initialize method. When it is called, the listener can free up any resources it uses.

See Also:

ObjectGrid.destroy()

---

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup

public interface **ObjectGridEventGroup**

This is a set of single method interfaces for fine grained events delivered for an ObjectGrid. Classes implementing these interfaces AND ObjectGridEventListener can receive these events. If an ObjectGridEventListener implements ANY of these interfaces that only the specific methods on the interfaces implemented will be called.

Since:

WAS XD 6.1, XC10

See Also:

ObjectGridEventListener

Nested Class Summary

static interface	**ObjectGridEventGroup.ShardEvents** These events are fired when a shard is made a primary shard and when the shard is demoted from a primary.
static interface	**ObjectGridEventGroup.ShardLifecycle** These events are fired when an ObjectGrid shard is initialized and destroyed.
static interface	**ObjectGridEventGroup.TransactionEvents** These events are called every single transaction.

e r f a c e	

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup.ShardLifecycle

Enclosing interface:

ObjectGridEventGroup

`public static interface` **`ObjectGridEventGroup.ShardLifecycle`**

These events are fired when an ObjectGrid shard is initialized and destroyed. A shard can be activated/deactivated multiple times within these two events.

Method Summary

void	**destroy**() Called when the `ObjectGrid` associated with this listener is destroyed.
void	**initialize**(Session session) Invoked when an `ObjectGrid` is initialized.

Method Detail

initialize

`void` **`initialize`**(Session session)

Invoked when an `ObjectGrid` is initialized.

A usable `Session` instance is passed to this listener to provide all of the necessary access to the various ObjectGrid objects.

Parameters:

`session` - a Session instance that this listener is associated with.

See Also:

ObjectGrid.initialize()

---

destroy

`void` **`destroy`**()

Called when the `ObjectGrid` associated with this listener is destroyed.

This method is the opposite of the `initialize` method. When it is called, the listener can free up any resources it uses.

See Also:

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup.ShardEvents

Enclosing interface:

ObjectGridEventGroup

`public static interface` **`ObjectGridEventGroup.ShardEvents`**

These events are fired when a shard is made a primary shard and when the shard is demoted from a primary.

Method Summary

void	**shardActivated**(ObjectGrid grid) This is called when a shard is promoted to a primary.
void	**shardDeactivate**(ObjectGrid grid) This is called when a primary shard is demoted to a replica.

Method Detail

shardActivated

`void` **`shardActivated`**(ObjectGrid grid)

This is called when a shard is promoted to a primary.

Parameters:

`grid` - This is a local reference to the shard containing the primary data.

---

shardDeactivate

`void` **`shardDeactivate`**(ObjectGrid grid)

This is called when a primary shard is demoted to a replica. This can happen is the balancer decides the primary is better placed in a different container. Replication is still active until this method returns to the caller. If any application controlled transactions are in flight then they should be stopped before returning. Once this method returns then any remaining transactions will fail.

Parameters:

`grid` - A reference to the shard.

---

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup.TransactionEvents

Enclosing interface:

ObjectGridEventGroup

`public static interface` **`ObjectGridEventGroup.TransactionEvents`**

These events are called every single transaction. These are primarily used when transaction level listening is required. This is usually for pushing changes or invalidation events to peer caches for simple scenarios.

Method Summary

void	**transactionBegin**(String txid, boolean isWriteThroughEnabled) Signals the beginning of a Session transaction.
void	**transactionEnd**(String txid, boolean isWriteThroughEnabled, boolean committed, Collection changes) Signals the ending of a Session transaction.

Method Detail

transactionBegin

```
void transactionBegin(String txid,
                      boolean isWriteThroughEnabled)
```

Signals the beginning of a Session transaction.

A stringified version of the TxID is provided for correlating with the end of the transaction, if so desired. The type of transaction is also provided by the isWriteThroughEnabled boolean parameter.

Parameters:

txid - Stringified version of the TxID

isWriteThroughEnabled - boolean flag indicating whether the Session transaction was started using the Session.beginNoWriteThrough(). method. false is passed if beginNoWriteThrough() was used.

See Also:

Session.begin(), Session.beginNoWriteThrough()

---

transactionEnd

```
void transactionEnd(String txid,
                    boolean isWriteThroughEnabled,
```

```
boolean committed,
Collection changes)
```

Signals the ending of a `Session` transaction.

A string version of the TxID is provided for correlating with the begin of the transaction, if so desired. Map changes are also reported with the collection of `LogSequence`s passed to this method. Typical uses of this event are for customers doing custom peer invalidation or peer commit push. This event listener gives them the changes. Calls to this method are made after commit and are sequenced so that they are delivered one by one, not in parallel. The event order is the commit and rollback order.

Parameters:

`txid` - string version of the TxID
`isWriteThroughEnabled` - boolean flag indicating whether the Session transaction was started using the `Session.beginNoWriteThrough()`. method. `false` is passed if `beginNoWriteThrough()` was used.
`committed` - a boolean flag indicating whether the transaction was committed (`true`) or rolled back (`false`)
`changes` - a Collection of LogSequences representing the changes that were committed or rolled back.

See Also:

`LogSequence.isRollback()`, `Session.begin()`, `Session.beginNoWriteThrough()`, `Session.commit()`, `Session.rollback()`

---

com.ibm.websphere.objectgrid.plugins

Interface MapEventListener

All Superinterfaces:

com.ibm.websphere.objectgrid.plugins.EventListener

```
public interface MapEventListener
extends com.ibm.websphere.objectgrid.plugins.EventListener
```

This callback interface is implemented by the application when it wants to receive events about a Map such as the eviction of a map entry.

Since:

WAS XD 6.0, XC10

See Also:

BackingMap.addMapEventListener(EventListener), BackingMap.removeMapEventListener(EventListener), EventListener

---

Method Summary

void	**entryEvicted**(Object key, Object value) Invoked when the specified entry is evicted from the map.
void	**preloadCompleted**(Throwable t) Invoked when the preloading of this map has completed.

Method Detail

entryEvicted

```
void entryEvicted(Object key,
                  Object value)
```

Invoked when the specified entry is evicted from the map.

The eviction could have occurred either by an `Evictor`'s processing or by invoking one of the `invalidate` methods on the `ObjectMap`.

For a MapEventListener in an `ObjectMap` that is configured to use OutputFormat.RAW for the keys and values, the keys and values objects passed will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the implementation class.

Parameters:

`key` - The key for the map entry that was evicted.
`value` - The value that was in in the map entry evicted. The value object should not be modified.

See Also:

Evictor, EvictionEventCallback, ObjectMap.invalidate(Object, boolean)

---

preloadCompleted

`void` **`preloadCompleted`**(Throwable `t`)

Invoked when the preloading of this map has completed.

This method is useful to determine when a preload operation finishes if asynchronous preloading is enabled. In addition if any error occurred during synchronous or asynchronous preload, it is reported with the invocation of this method.

Parameters:

`t` - A Throwable object that indicates if preload completed without any Throwable occuring during the preload of the map. A null reference indicates preload completed without any Throwable objects occuring during the preload of the map.

See Also:

`Loader.preloadMap(Session, BackingMap)`, BackingMap.setPreloadMode(boolean)

---

com.ibm.websphere.objectgrid.plugins

Interface LogSequenceFilter

public interface **LogSequenceFilter**

This interface can be used to filter a LogSequence. As an operation, such as serialization, needs to know whether a given LogElement should be included or not, this callback object will be used for the boolean check. If the given LogElement should be used in the operation, then "true" should be returned. If the given LogElement should not be used, then "false" should be returned. This interface is primarily used by the `serialize` method of the `LogSequenceTransformer` class.

Since:

WAS XD 6.0, XC10

Method Summary

boolean	**accept**(LogElement logElement) Returns true if the given LogElement should be used; false if the given LogElement should not be used.

Method Detail

accept

boolean **accept**(LogElement logElement)

Returns true if the given LogElement should be used; false if the given LogElement should not be used.

Parameters:

`logElement` - the LogElement to be filtered

Returns:

true if the given LogElement should be used in the operation; false otherwise.

com.ibm.websphere.objectgrid.plugins

Interface LogSequence

All Superinterfaces:

Serializable

```
public interface LogSequence
extends Serializable
```

LogSequence is the ordered list of changes performed against a given map for a given transaction. These changes are recorded as LogElement objects.

Since:

WAS XD 6.0, XC10

Method Summary

Iterator	**getAllChanges**() Returns an iterator for processing all of the changes for a LogSequence.
Iterator	**getChangesByKeys**(Collection keys) Returns an iterator for processing the LogElements that have the requested keys.
Iterator	**getChangesByTypes**(Collection types) Returns an iterator for processing the LogElements that are of the requested type.
String	**getMapName**() Returns the name of the map that these changes apply to.
String	**getObjectGridName**() Returns the name of the ObjectGrid that houses the map that these changes apply to.

Iterator	**getPendingChanges**() Returns an iterator for processing all of the "pending" changes for a LogSequence (for example, pending inserts, updates, and deletes).
boolean	**isDirty**() Returns whether this LogSequence has any LogElements that would "dirty" a Map.
boolean	**isRollback**() Returns whether or not this LogSequence was generated to rollback a transaction.
int	**size**() Returns the total number of LogElements within this LogSequence.

Method Detail

size

`int` **`size`**`()`

Returns the total number of LogElements within this LogSequence.

Returns:
total number of LogElements

---

getPendingChanges

`Iterator` **`getPendingChanges`**`()`

Returns an iterator for processing all of the "pending" changes for a LogSequence (for example, pending inserts, updates, and deletes).

This method is normally used by a `Loader`. A pending change is one that has not been written out to a loader yet using a `flush()` operation. Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Returns:
an Iterator for processing the pending LogElement changes
See Also:
`ObjectMap.flush()`, `Session.flush()`

---

getAllChanges

`Iterator` **`getAllChanges`**`()`

Returns an iterator for processing all of the changes for a LogSequence.

This method would normally be used by an `Evictor` and other plugins that want to know all of the changes introduced by this LogSequence. Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Returns:
an Iterator for processing all of the LogElement changes

---

getChangesByTypes

`Iterator` **`getChangesByTypes`**`(Collection types)`

Returns an iterator for processing the LogElements that are of the requested type. Each member of the input Collection should be one of the defined LogElement Types (INSERT, UPDATE, DELETE, FETCH, TOUCH, or EVICT). Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Parameters:
`types` - A Collection of LogElement Types (INSERT, UPDATE, etc)
Returns:
Iterator for processing all LogElements that support the input Type(s)
Throws:
`IllegalArgumentException` - if types is `null`
See Also:
`LogElement.DELETE`, `LogElement.EVICT`, `LogElement.FETCH`, `LogElement.INSERT`, `LogElement.TOUCH`, `LogElement.UPDATE`, `LogElement.CLEAR`

---

getChangesByKeys

`Iterator` **`getChangesByKeys`**`(Collection keys)`

Returns an iterator for processing the LogElements that have the requested keys.

Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Parameters:
`keys` - a collection of key objects
Returns:
an Iterator for processing all LogElements that match the input key(s)

---

getMapName

`String` **`getMapName`**`()`

Returns the name of the map that these changes apply to.

The caller can use the return value of this method as input to the `Session.getMap(String)` method.

Returns:
The name of the map that these changes apply to
See Also:
`Session.getMap(String)`

---

getObjectGridName

`String` **`getObjectGridName`**`()`

Returns the name of the ObjectGrid that houses the map that these changes apply to.

Returns:
The name of the ObjectGrid that this LogSequence is associated with
Since:
WAS XD 6.0.1

---

isDirty

`boolean` **`isDirty`**`()`

Returns whether this LogSequence has any LogElements that would "dirty" a Map.
That is, if it contains any LogElements of any type other than Fetch/Get, it is considered "dirty".

Returns:
`true` if the LogSequence would modify a Map, if applied; `false` if the LogSequence would not modify a Map, if applied

---

isRollback

`boolean` **`isRollback`**`()`

Returns whether or not this LogSequence was generated to rollback a transaction.

Note, depending on when this LogSequence is used, the transaction itself might already be rolled back.

Returns:
true iff this LogSequence was generated to rollback a transaction.
Since:
WAS XD 6.0.1

---

com.ibm.websphere.objectgrid.plugins

Interface LogElement

All Superinterfaces:

Serializable

```
public interface LogElement
extends Serializable
```

LogElements are the individual entries within a LogSequence. A LogElement has attributes such as operation type (delete, insert, update, etc.), current value, last access time, versioned value, etc. A LogElement is created during a transaction to record in-flight operations. For a LogElement on an `ObjectMap` that is configured to use OutputFormat.RAW for the keys or values, the keys or values objects in the LogElement will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object. To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

Since:

WAS XD 6.0, XC10

See Also:

LogSequence

Nested Class Summary

static class	**LogElement.Type** The `Type` class is used to represent a `LogElement` type.

Field Summary

static LogElement	**CLEAR** The `Type` that represents the CLEAR operation.

[t](#).[T](#)[y](#)[p](#)[e](#)	
static int	**CODE_CLEAR** The type code constant for `CLEAR`.
static int	**CODE_DELETE** The type code constant for `DELETE`.
static int	**CODE_EVICT** The type code constant for `EVICT`.
static int	**CODE_FETCH** The type code constant for `FETCH`.
static int	**CODE_INSERT** The type code constant for INSERT.
static int	**CODE_LOCK** The type code constant for `LOCK`.
static int	**CODE_TOUCH** The type code constant for `TOUCH`.

int	
static int	**CODE_UNDO_NOT_NEEDED** The code constant for UNDO_NOT_NEEDED.
static int	**CODE_UPDATE** The type code constant for UPDATE.
static int	**CODE_UPSERT** The type code constant for UPSERT.
static LogElement.Type	**DELETE** The Type that represents the DELETE operation.
static LogElement.Type	**EVICT** The Type that represents the EVICT operation.

static LogElement.Type	**FETCH** The Type that represents the FETCH operation.
static LogElement.Type	**INSERT** The Type that represents the INSERT operation.
static LogElement.Type	**LOCK** The Type that represents the LOCK operation.
static LogElement.Type	**TOUCH** The Type that represents the TOUCH operation.

[ent.Type](#)	
static [LogElement.Type](#)	**UNDO_NOT_NEEDED** The `Type` that represents an UNDO action is NOT required for this `LogElement`.
static [LogElement.Type](#)	**UPDATE** The `Type` that represents the UPDATE operation.
static [LogElement.Type](#)	**UPSERT** The `Type` that represents the UPSERT operation.

Method Summary

Object	**getAfterImage**() Gets the "after image" value object.
Object	**getBeforeImage**() Gets the "before image" of the value object.
CacheEntry	**getCacheEntry**() Returns the `CacheEntry` for this key.
Object	**getCurrentValue**() Gets the value for this `LogElement`.
Object	**getKey**() Returns the key for this LogElement.
long	**getLastAccessTime**() Returns the last access time associated with this `LogElement`.
LogElement.Type	**getType**() Gets the type of this `LogElement`.
LogElement.Type	**getUndoType**() Returns what operation must be performed to "undo" a prior change the transaction made to the map entry.

[e](#)	
[Object](#)	**getVersionedValue**() Gets the versioned object at the time the object was first associated with the transaction.
boolean	**isCascaded**() Answers `true` if this LogElement is a result of a cascade operation.
boolean	**isPending**() Answers `true` if this change has NOT been applied to the loader.
void	**setVersionedValue**([Object](#) v) Used to update the versioned object after an update of map entry occurs.

Field Detail

CODE_INSERT

```
static final int CODE_INSERT
```

The type code constant for INSERT.

See Also:

[INSERT](#), [LogElement.Type.getCode()](#), Constant Field Values

---

CODE_UPDATE

```
static final int CODE_UPDATE
```

The type code constant for `UPDATE`.

See Also:

[UPDATE](#), [LogElement.Type.getCode()](#), Constant Field Values

---

CODE_DELETE

```
static final int CODE_DELETE
```

The type code constant for `DELETE`.

See Also:

[DELETE](#), [LogElement.Type.getCode()](#), Constant Field Values

---

CODE_EVICT

`static final int` **`CODE_EVICT`**

The type code constant for `EVICT`.

See Also:
`EVICT`, `LogElement.Type.getCode()`, Constant Field Values

CODE_FETCH

`static final int` **`CODE_FETCH`**

The type code constant for `FETCH`.

See Also:
`FETCH`, `LogElement.Type.getCode()`, Constant Field Values

CODE_TOUCH

`static final int` **`CODE_TOUCH`**

The type code constant for `TOUCH`.

See Also:
`TOUCH`, `LogElement.Type.getCode()`, Constant Field Values

CODE_CLEAR

`static final int` **`CODE_CLEAR`**

The type code constant for `CLEAR`.

Since:
WAS XD 6.1.0.3
See Also:
`CLEAR`, `LogElement.Type.getCode()`, Constant Field Values

CODE_LOCK

`static final int` **`CODE_LOCK`**

The type code constant for `LOCK`.

Since:
8.6, XC10 2.5
See Also:
`LOCK`, `LogElement.Type.getCode()`, Constant Field Values

CODE_UPSERT

`static final int` **`CODE_UPSERT`**

The type code constant for `UPSERT`.

Since:
8.6, XC10 2.5
See Also:
`UPSERT`, `LogElement.Type.getCode()`, Constant Field Values

CODE_UNDO_NOT_NEEDED

`static final int` **`CODE_UNDO_NOT_NEEDED`**

The code constant for `UNDO_NOT_NEEDED`. Used to indicate no operation is needed to undo the changes for this `LogElement` since this `LogElement` was never processed or it was an operation that does not require an undo operation.

See Also:
Constant Field Values

INSERT

`static final LogElement.Type` **`INSERT`**

The `Type` that represents the INSERT operation.

UPDATE

`static final LogElement.Type` **`UPDATE`**

The `Type` that represents the UPDATE operation.

DELETE

`static final LogElement.Type` **`DELETE`**

The `Type` that represents the DELETE operation.

EVICT

`static final LogElement.Type` **`EVICT`**

The `Type` that represents the EVICT operation.

FETCH

`static final LogElement.Type` **`FETCH`**

The `Type` that represents the FETCH operation.

TOUCH

`static final LogElement.Type` **`TOUCH`**

The `Type` that represents the TOUCH operation.

CLEAR

`static final LogElement.Type` **`CLEAR`**

The `Type` that represents the CLEAR operation.

Since:
WAS XD 6.1.0.3

LOCK

static final LogElement.Type **LOCK**

The Type that represents the LOCK operation.

Since:
8.6, XC10 2.5

UPSERT

static final LogElement.Type **UPSERT**

The Type that represents the UPSERT operation.

Since:
8.6, XC10 2.5

UNDO_NOT_NEEDED

static final LogElement.Type **UNDO_NOT_NEEDED**

The Type that represents an UNDO action is NOT required for this LogElement.

Method Detail

getType

LogElement.Type **getType**()

Gets the type of this LogElement. The type indicates what operation needs to be applied to the map entry.

Returns:
the type of this LogElement.

---

getCurrentValue

Object **getCurrentValue**()

Gets the value for this LogElement.

For a LogElement on an ObjectMap that is configured to use a ValueSerializerPlugin, the values in the LogSequence will be SerializedValue objects. If required, you can use the SerializedEntry.getObject() method to retrieve (possibly inflating the serialized object) the original value object.

The original value represents the new value that should be applied to the BackingMap and Loader. This value can be cast to ValueProxyInfo when a value interface is in use in order to determine the set of dirty attributes.

Returns:
the value in case of INSERT, UPDATE, UPSERT, or FETCH, null in the case of DELETE or EVICT.

See Also:
ValueProxyInfo

---

getCacheEntry

`CacheEntry` **`getCacheEntry`**`()`

Returns the `CacheEntry` for this key. The key, current committed value, etc. can be accessed from the `CacheEntry`.

Returns:
the entry in the cache that is requested to be updated.
See Also:
`CacheEntry.getCommittedValue()`, `getKey()`

---

isPending

`boolean` **`isPending`**`()`

Answers `true` if this change has NOT been applied to the loader.

Changes can previously be applied to a loader using the `ObjectMap.flush()` or `Session.flush()` methods. This method reveals whether the change in this `LogElement` has already been applied to the Loader using one of those methods.

Returns:
`true` if this change has NOT been applied to the loader.
See Also:
`ObjectMap.flush()`, `Session.flush()`

---

getVersionedValue

`Object` **`getVersionedValue`**`()`

Gets the versioned object at the time the object was first associated with the transaction.

For a LogElement on an `ObjectMap` that is configured to use a or `ValueSerializerPlugin`, the versioned object will be returned as an `XsDataInputStream`, read will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object. For a LogElement on an `ObjectMap` that is configured to use a `ValueSerializerPlugin` that generates version objects, the version object will be the data stream representing the data.

Returns:
The versioned object.
See Also:
`OptimisticCallback`

---

setVersionedValue

`void` **`setVersionedValue`**`(``Object`` v)`

Used to update the versioned object after an update of map entry occurs.

The Loader can use this method when it is using an optimistic strategy and uses the `OptimisticCallback.updateVersionedObjectForValue(Object)` method to get an updated version object.

Parameters:
`v` - The versioned object.
See Also:
`OptimisticCallback.updateVersionedObjectForValue(Object)`

getLastAccessTime

`long` **`getLastAccessTime`**`()`

Returns the last access time associated with this `LogElement`.

Returns:
last access time

---

getUndoType

`LogElement.Type` **`getUndoType`**`()`

Returns what operation must be performed to "undo" a prior change the transaction made to the map entry.

Note, an undo type of `UNDO_NOT_NEEDED` is returned if nothing needs to be undone for this `LogElement`.

Returns:
the "undo" type of this `LogElement`. It can be one of: INSERT, UPDATE, DELETE or UNDO_NOT_NEEDED

---

getBeforeImage

`Object` **`getBeforeImage`**`()`

Gets the "before image" of the value object.

The "before image" is the value object that existed in map entry prior to applying a change to map entry. Note, it is possible for a `null` reference to be returned (e.g. in the case where a new map entry is created).

For a LogElement on an `ObjectMap` that is configured to use `OutputFormat.RAW` for the values, the value will be a `SerializedValue` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

Returns:
the value prior to applying the change

---

getAfterImage

`Object` **`getAfterImage`**`()`

Gets the "after image" value object.

The "after image" is the value object that existed in map entry after applying a change to the map entry. Note, it is possible for a `null` reference to be returned (e.g. in the case where an existing map entry is removed/evicted).

For a LogElement on an `ObjectMap` that is configured to use `OutputFormat.RAW` for the value, the value will be a `SerializedValue` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

Returns:
the value after applying the change

---

isCascaded

`boolean` **`isCascaded`**`()`

Answers `true` if this LogElement is a result of a cascade operation. This only applies to ObjectGrid EntityManager programming model.

ObjectGrid EntityManager supports cascade operations. For example, when persisting an entity P, if P has a relation to entity C with `CascadeType.PERSIST` enabled, C will also be persisted as a result of the cascade operation. The method isCascaded() returns true for the LogElement object which represents C, and the method returns false for the LogElement object which represents P.

Returns:
`true` if the LogElement object is a result of cascade operation.
Since:
6.1.0.5 FIX1
See Also:
`EntityManager`

---

getKey

`Object` **`getKey`**`()`

Returns the key for this LogElement.

For a LogElement on an `ObjectMap` that is configured to use `OutputFormat.RAW` for the keys, the value will be a `SerializedKey` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

This method can be used instead of LogElement.getCacheEntry().getKey().

Returns:
the key for this LogElement.
Since:
7.0
See Also:
`CacheEntry.getKey()`

---

com.ibm.websphere.objectgrid.plugins

Class LogElement.Type

java.lang.Object
  com.ibm.websphere.objectgrid.plugins.LogElement.Type

All Implemented Interfaces:
Comparable

Enclosing interface:
LogElement

```
public static class LogElement.Type
extends Object
implements Comparable
```

The `Type` class is used to represent a `LogElement` type.

Since:
WAS XD 6.0

Method Summary

int	**compareTo**(Object object)
int	**getCode**() Gets the type code for this object.
String	**toString**()
static LogElement.Type	**valueOf**(int typeCode) Return the Type for a given type code.

p e	

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Method Detail

getCode

```
public int getCode()
```

Gets the type code for this object.

Returns:
the type code

---

compareTo

```
public int compareTo(Object object)
```

Specified by:
compareTo in interface Comparable

See Also:
Comparable.compareTo(Object)

---

toString

```
public String toString()
```

Overrides:
toString in class Object

See Also:
Object.toString()

---

valueOf

```
public static LogElement.Type valueOf(int typeCode)
```

Return the Type for a given type code.

Parameters:
typeCode - the typecode
Returns:
the Type
Throws:
IllegalArgumentException - if the typeCode isn't valid.
Since:
8.6, XC10 2.5

---

com.ibm.websphere.objectgrid.plugins

Class LoaderException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

UnavailableServiceException

---

public class **LoaderException**
extends ObjectGridException

This exception is the base exception for any exceptions encountered by a `Loader`.

Since:

WAS XD 6.0, XC10

See Also:

`Loader`, Serialized Form

Constructor Summary

LoaderException() Constructs a new LoaderException with `null` as its detail message.
LoaderException(String message) Constructs a new LoaderException with the specified detail message.
LoaderException(String message, Throwable cause) Constructs a new LoaderException with the specified detail message and cause.
LoaderException(Throwable cause) Constructs a new LoaderException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LoaderException

public **LoaderException**()

Constructs a new LoaderException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

LoaderException

public **LoaderException**(String message)

Constructs a new LoaderException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
ObjectGridException.initCause(Throwable), Throwable.getMessage()

LoaderException

public **LoaderException**(String message,
Throwable cause)

Constructs a new LoaderException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LoaderException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
ObjectGridException.getCause(), Throwable.getMessage()

LoaderException

public **LoaderException**(Throwable cause)

Constructs a new LoaderException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LoaderExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for

later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid.plugins

Interface EvictorData

All Known Implementing Classes:

EvictorData.SpecialEvictorData

`public interface` **`EvictorData`**

This interface is optionally used by an implementator of the `Evictor` interface. Application changes applied to a `BackingMap` are asynchronous from the `Evictor` activity. The `Evictor` is not notified of changes to the `BackingMap` until after application transactions are committed. Consequently, if an `Evictor` decides to evict a map entry, it is possible that the `BackingMap` could evict an entry that was different from the original entry it was tracking. For example, consider that an application could execute a transaction that removes a map entry. Before the `Evictor` is notified of the remove, another transaction inserts a new entry into the `BackingMap` for the same key as the old entry. Consequently, the `Evictor` could evict the newly created entry when it meant to evict the old entry. To help close this small timing window, the `Evictor` can use this interface to associate evictor specific data with a map entry. The `Evictor` can then do the following:

- store the `EvictorData` object for a map entry by using the `EvictionEventCallback.setEvictorData(Object, EvictorData)` method.
- retrieve the `EvictorData` object for a map entry by using the `EvictionEventCallback.getEvictorData(Object)` method.
- Conditionally evict a map entry if and only if the cache entry for a specified key has the exact same `EvictorData` object (the java == operator returns true) associated with it by using the `EvictionEventCallback.evictMapEntries(List)` method.

Since:
WAS XD 6.0.1, XC10

See Also:
`Evictor`, `EvictionEventCallback`

Nested Class Summary

static class	**EvictorData.SpecialEvictorData** Special value class used for representing the key not being found in the `BackingMap`.

Field Summary

static	

static [EvictorData](#)	**KEY_NOT_FOUND** A special value indicating that the key was not found.

Method Summary

[Object](#)	**getKey**() Retrieves the key object for this `EvictorData` instance.

Field Detail

KEY_NOT_FOUND

`static final EvictorData` **KEY_NOT_FOUND**

A special value indicating that the key was not found.

Method Detail

getKey

`Object` **getKey**()

Retrieves the key object for this `EvictorData` instance.

Returns:

the same key object that was passed to the `EvictionEventCallback.setEvictorData(Object, EvictorData)` method when this EvictorData was associated with the map entry with the given key.

See Also:

`EvictionEventCallback.setEvictorData(Object, EvictorData)`

com.ibm.websphere.objectgrid.plugins

Class EvictorData.SpecialEvictorData

java.lang.Object

  com.ibm.websphere.objectgrid.plugins.EvictorData.SpecialEvictorData

All Implemented Interfaces:

EvictorData

Enclosing interface:

EvictorData

```
public static final class EvictorData.SpecialEvictorData
extends Object
implements EvictorData
```

Special value class used for representing the key not being found in the `BackingMap`.

Since:

WAS XD 6.0.1

Nested Class Summary

Nested classes/interfaces inherited from interface com.ibm.websphere.objectgrid.plugins.EvictorData
EvictorData.SpecialEvictorData

Field Summary

Fields inherited from interface com.ibm.websphere.objectgrid.plugins.EvictorData
KEY_NOT_FOUND

Constructor Summary

EvictorData.SpecialEvictorData()

Method Summary

Object	**getKey**() Dummy implementation method since this class will not be called.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EvictorData.SpecialEvictorData

public **EvictorData.SpecialEvictorData**()

Method Detail

getKey

public Object **getKey**()

Dummy implementation method since this class will not be called.

Specified by:
getKey in interface EvictorData

Returns:
null

See Also:
EvictionEventCallback.setEvictorData(Object, EvictorData)

---

com.ibm.websphere.objectgrid.plugins

Interface Evictor

All Known Implementing Classes:

LFUEvictor, LRUEvictor

`public interface` **`Evictor`**

Data contained in a `BackingMap` are evicted when the map is full. This plugin is used by the `BackingMap` to determine when and what to evict from the map based on some algorithm (LRU, LFU, time based, etc).

An `Evictor` implementation that also implements the `BackingMapLifecycleListener` interface will be automatically added as an `EventListener` on the `BackingMap` when the evictor set on the backing map.

An `Evictor` may also implement the `BackingMapPlugin` interface in order to receive enhanced BackingMap plug-in lifecycle method calls. The plug-in is then also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Since:

WAS XD 6.0, XC10

See Also:

BackingMap.addMapEventListener(EventListener), BackingMap.setEvictor(Evictor), EvictorData

---

Method Summary

void	**activate**() This method is called to activate the Evictor.
void	**apply**(LogSequence sequence) Called after a transaction has committed to allow the evictor to track object usage in the `BackingMap`.
void	**deactivate**() This method is called to deactivate the Evictor.
void	**destroy**() Called when the `BackingMap` associated with this evictor is destroyed.
void	**initialize**(BackingMap map, EvictionEventCallback callback) Called by a `BackingMap` instance during the evictor initialization time.

Method Detail

initialize

```
void initialize(BackingMap map,
                EvictionEventCallback callback)
```

Called by a `BackingMap` instance during the evictor initialization time.

The `BackingMap` calls this method so the Evictor instance can have references to the `BackingMap` and `EvictionEventCallback` instances. The evictor can signal events to have specific entries evicted using the `EvictionEventCallback`.

Parameters:

`map` - the BackingMap instance
`callback` - the EvictionEventCallback instance

See Also:

BackingMap, EvictionEventCallback

---

destroy

```
void destroy()
```

Called when the `BackingMap` associated with this evictor is destroyed.

This method is the opposite of the `initialize` method. When it is called, the Evictor can free up any resources it uses.

See Also:

ObjectGrid.destroy()

---

apply

```
void apply(LogSequence sequence)
```

Called after a transaction has committed to allow the evictor to track object usage in the `BackingMap`.

This method also reports any entries that have been successfully evicted. Note, this method is not called for transactions that are rolled back. If there is a need to track object usage for rolled back transactions, the evictor must implement the `RollbackEvictor` interface as well.

This method is called after a transaction has completed. Consequently, all transaction locks that were acquired by the completed transaction are no longer held. Potentially, multiple threads could call this method concurrently and each thread would be completing its own transaction. Since transaction locks are already released by the completed transaction, this method must provide its own synchronization to ensure it is thread safe. For an Evictor in an `ObjectMap` that is configured to use `OutputFormat.RAW` for the keys or values, the keys and values objects in the LogSequence will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object. To override the map's output format configuration, use the `PluginOutputFormat` annotation in the implementation class.

Parameters:

`sequence` - the `LogSequence` of changes committed to the map

See Also:

`RollbackEvictor`

---

activate

`void **activate**()`

This method is called to activate the Evictor. Until this method is called, the Evictor must not use the EvictionEventCallback interface to evict any map entries. If it does use the EvictionEventcallback interface to evict map entries prior to activate being called, an IllegalStateException is thrown.

---

deactivate

`void **deactivate**()`

This method is called to deactivate the Evictor. Once this method is called, the Evictor must quit using the EvictionEventCallback interface to evict any map entries. If it does use the EvictionEventcallback interface after this method is called, an IllegalStateException is thrown.

---

com.ibm.websphere.objectgrid.plugins

Interface EvictionEventCallback

public interface **EvictionEventCallback**

An instance of `EvictionEventCallback` is passed into the `Evictor` at initialization time. When an eviction method is called, corresponding methods of `EvictionEventCallback` will be called so the `BackingMap` can process evictions.

Since:

WAS XD 6.0, XC10

See Also:

`Evictor`, `EvictorData`

Method Summary

void	**evictEntries**(List keysToEvictList) If an `Evictor` chooses not to implement the `EvictorData` interface, this method can be used to evict a map entry.
void	**evictMapEntries**(List evictorDataList) This method is the preferred method for the `Evictor` to use when evicting map entries.
EvictorData	**getEvictorData**(Object key) Gets the evictor data for a specified BackingMap cache entry.
void	**setEvictorData**(Object key, EvictorData data) Sets the evictor data for a specified BackingMap cache key.

Method Detail

setEvictorData

void **setEvictorData**(Object key,
EvictorData data)

Sets the evictor data for a specified BackingMap cache key.

This method can be used by an implementor of the `Evictor` interface to keep data that the evictor needs for determining which cache entry to evict.

Parameters:

`key` - is the key for accessing a BackingMap entry.

`data` - the EvictorData object to store as evictor data for a specified key.

Throws:

`IllegalArgumentException` - if key is a null reference or there is no BackingMap cache entry for this key.

Since:

WAS XD 6.0.1

See Also:

`Evictor`

---

getEvictorData

`EvictorData` **`getEvictorData`**`(Object key)`

Gets the evictor data for a specified BackingMap cache entry.

Parameters:

`key` - the key for the BackingMap entry to set.

Returns:

if the specified key is not found in BackingMap, then the special value `EvictorData.KEY_NOT_FOUND` is returned. If the key is found in the BackingMap, the same reference that was previously passed to the `setEvictorData(Object, EvictorData)` method of this interface is returned. A null reference is returned if the key is found, but the `setEvictorData` method was not previously called for the specified key.

Throws:

`IllegalArgumentException` - if key is a null reference.

Since:

WAS XD 6.0.1

See Also:

`setEvictorData(Object, EvictorData)`, `EvictorData.KEY_NOT_FOUND`

---

evictMapEntries

```
void evictMapEntries(List evictorDataList)
                     throws ObjectGridException
```

This method is the preferred method for the `Evictor` to use when evicting map entries. A list of `EvictorData` objects is passed as an argument to this method. For each `EvictorData` object in the list, the key is obtained from the `EvictorData` object and used to determine which BackingMap entry to evict. The BackingMap entry is evicted if and only if the cache entry for BackingMap entry contains the exact same `EvictorData` object in it. That is, the java == operator is used to ensure it is the exact same `EvictorData` object. If the == operator indicates a different object, then the map entry is not evicted. For those map entries that are physically evicted from the map, the `Evictor` will receive notification through its `apply` method.

Parameters:

`evictorDataList` - a list of `EvictorData` objects to process. The caller must guarantee this parameter is not `null` or contain any `null` references.

Throws:

`ObjectGridException` - if an error occurs during processing

`ClassCastException` - if an object in evictorDataList does not implement the `EvictorData` interface.

Since:

WAS XD 6.0.1

See Also:

Evictor.apply(LogSequence), EvictorData.getKey()

---

evictEntries

```
void evictEntries(List keysToEvictList)
                  throws ObjectGridException
```

If an `Evictor` chooses not to implement the `EvictorData` interface, this method can be used to evict a map entry. However, the `Evictor` must be prepared to handle the exposure of an application removing and recreating a map entry before the `Evictor` has an opportunity to call this method.

For this method, a list of map keys is passed. The list is evaluated and an eviction is conducted on the list. When the entries are physically evicted from the map, the `Evictor` will receive notification through its `apply` method.

Parameters:

`keysToEvictList` - List of keys to evict from the map. The caller must guarantee this parameter is not `null` or contain any `null` references.

Throws:

`ObjectGridException` - if an error occurs during processing

See Also:

`Evictor.apply(LogSequence)`

---

com.ibm.websphere.objectgrid.plugins

Class CacheEntryException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.CacheEntryException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **CacheEntryException**
extends ObjectGridException

This exception indicates an error occurred during a cache entry operation.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

CacheEntryException() Constructs a new CacheEntryException with `null` as its detail message.
CacheEntryException(String message) Constructs a new CacheEntryException with the specified detail message.
CacheEntryException(String message, Throwable cause) Constructs a new CacheEntryException with the specified detail message and cause.
CacheEntryException(Throwable cause) Constructs a new CacheEntryException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CacheEntryException

`public` **`CacheEntryException`**`()`

Constructs a new CacheEntryException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

CacheEntryException

`public` **`CacheEntryException`**`(String message)`

Constructs a new CacheEntryException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

CacheEntryException

`public` **`CacheEntryException`**`(String message,`
`Throwable cause)`

Constructs a new CacheEntryException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this CacheEntryException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

CacheEntryException

`public` **`CacheEntryException`**`(Throwable cause)`

Constructs a new CacheEntryException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for CacheEntryExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.
See Also:
`ObjectGridException.getCause()`

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

com.ibm.websphere.objectgrid.plugins

Interface CacheEntry

All Superinterfaces:

Serializable

```
public interface CacheEntry
extends Serializable
```

This interface represents a cache entry in an ObjectGrid map.

Since:

WAS XD 6.0, XC10

Method Summary

Object	**getCommittedValue**() Returns the committed value for this entry.
Object	**getKey**() Returns the key for this entry.
long	**getLastAccessTime**() Returns the last time this entry was accessed.
long	**getTTL**() Returns the time-to-live value for this entry.
boolean	**isInBackingMap**() Indicates whether this entry is in the `BackingMap`

Method Detail

isInBackingMap

```
boolean isInBackingMap()
```

Indicates whether this entry is in the `BackingMap`

Returns:
Returns `true` if this element is in the BackingMap

---

getKey

`Object` **`getKey`**`()`

Returns the key for this entry.

For a CacheEntry on an `ObjectMap` that is configured to use a `KeySerializerPlugin`, the value will be a `SerializedKey` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key object.

Returns:
the key

---

getCommittedValue

`Object` **`getCommittedValue`**`()`

Returns the committed value for this entry.

The type of the object returned from the `getCommittedValue()` method depends on various configuration and storage options used by the `ObjectMap` that holds the CacheEntry. In the default case, `getCommittedValue()` returns the Java object of the same type that was put into the map. For an `ObjectMap` that is configured to use a `ValueSerializerPlugin`, the committed value depends on the underlying storage mechanism, typically represented as an array of bytes.

Returns:
the committed value

---

getTTL

`long` **`getTTL`**`()`

Returns the time-to-live value for this entry.

Returns:
the time-to-live value

---

getLastAccessTime

`long` **`getLastAccessTime`**`()`

Returns the last time this entry was accessed.

Returns:
last access time.

---

com.ibm.websphere.objectgrid.plugins

Interface BeanFactory

public interface **BeanFactory**

Implement this interface to allow bean factories like Spring or Google guice to be integrated. This allows ObjectGrid to delegate to an external Bean Factory to instantiate beans needed by ObjectGrid.

Since:

WAS XD 6.1 FIX3, XC10

Method Summary

Object	**getBean**(String name) This returns an instance of the bean with the specified name.

Method Detail

getBean

Object **getBean**(String name)

This returns an instance of the bean with the specified name.

Parameters:

name - The name of the bean instance to return.

Returns:

the bean instance.

Package com.ibm.websphere.objectgrid.security

This package has the class MapPermission and class AdminPermission which represents the permissions for to access the ObjectGrid maps and ObjectGrid administration respectively.

See:

Description

Class Summary	
SecurityConstants	This class contains the constants used for security configuration.

Exception Summary	
ObjectGridSecurityException	This exception represents a general ObjectGrid security exception.

Package com.ibm.websphere.objectgrid.security Description

This package has the class MapPermission and class AdminPermission which represents the permissions for to access the ObjectGrid maps and ObjectGrid administration respectively.

MapPermission action types.

The ObjectGrid defines 5 permission actions that are used to authorize accesses to the maps. These permissions allow access to maps to be controlled by an administrator. Objects within the ObjectGrid use a simple naming scheme. Each Map is named using the convention of the ObjectGrid name followed by a period followed by the Map name. For example, if the object grid name is "myObjectGrid" and the map name is "myMap", then the map name used in the permission is "myobjectgrid.mymap".

Wildcards can be used on names with some restrictions. A wild card "*" can be used to replace the map name or the object grid name, but not partially. For example, "myObjectGrid.*", "*.myMap", and "*.*" are valid names, but "myObject*.*" is not valid.

There are five actions with the permission object ObjectMapPermission.

- Read
  This action allows get operations to be issued against a Map.
- Write
  This action allows put operations to be issued against a Map. It allows existing entries to be updated.
- Remove
  This action allows entries to be removed from the Map.
- Insert
  This action allows clients to add entries to a Map.

- Invalidate
  This action allows clients to invalidate entries from the Map.

	com.ibm.websphere.objectgrid.ObjectMap/ com.ibm.websphere.objectgrid.JavaMap
Read	boolean containsKey(Object)
	boolean equals(Object)
	Object get(Object)
	Object get(Object, Serializable)
	List getAll(List)
	List getAll(List keyList, Serializable)
	List getAllForUpdate(List)
	List getAllForUpdate(List, Serializable)
	Object getForUpdate(Object)
	Object getForUpdate(Object, Serializable)
write	Object put(Object key, Object value)
	void put(Object, Object, Serializable)
	void putAll(Map)
	void putAll(Map, Serializable)
	void update(Object, Object)
	void update(Object, Object, Serializable)
insert	public void insert(Object, Object)
	void insert(Object, Object, Serializable)
	remove Object remove(Object)
	void removeAll(Collection)
invalidate	public void invalidate(Object, boolean)
	void invalidateAll(Collection, boolean)
	int setTimeToLive(int)

An authroizationMechanism setting of the ObjectGrid has two possible values: JAAS and custom. Users can also use API `ObjectGrid.setAuthorizationMechanism(int)` to set which authorization mechanism the object grid will use.

A value "JAAS" means ObjectGrid will rely on JAAS authorization mechanism to handle the authorization. A JAAS policy file should be configured to associate permissions with a set of credentials and/or groups of credentials. We recommend that groups should be used as then new users can be added to groups without modifying the policy file.

A value "custom" means ObjectGrid will rely on custom authorization mechanism to handle the authorization. Users can set call `ObjectGrid.setObjectGridAuthorization(com.ibm.websphere.objectgrid.security.plugins.ObjectGridAuthorization ogAuthorization)` to set their custom authorization plug-in. Users can also configure the objectgrid.xml to achieve the same result.

AdminPermission types

An AdminPermission has two types: ADMIN and MONITOR. An AdminPermission with ADMIN name grants permissions to access all the ManagementMBean methods. An AdminPermission with MONITOR name grants permissions to access the ManagementMBean read-only methods. Therefore, ADMIN permission implies MONITOR permission.

The detailed operations granted to users with different permissions are listed in the following

table. These operations correspond to the methods in the ManagementMBean interface:

operations	admin	monitor
startServer	Y	N
stopServer	Y	N
forceStopServer	Y	N
setServerTrace	Y	N
retrieveServerStatus	Y	Y
getMapStats	Y	Y
getOGStats	Y	Y
getReplicationStats	Y	Y

The table can read like this: If the client has admin permission, it can execute "startServer" task; if the client has monitor permission, it cannot execute "startServer" task.

AgentPermission types

An AgentPermission represents permissions to the datagrid agents. The name of the permission is the full name of the ObjectGrid map, and the action is a "," delimited string of agent implementation class names or package names.

The following methods in the class `AgentManager` requires AgentPermission:

- `AgentManager.callMapAgent(MapGridAgent, Collection)`
- `AgentManager.callMapAgent(MapGridAgent)`
- `AgentManager.callReduceAgent(ReduceGridAgent, Collection)`
- `AgentManager.callReduceAgent(ReduceGridAgent, Collection)`

ObjectGridPermission types

An ObjectGridPermission represents permissions to an ObjectGrid. The name of the permission is the ObjectGrid name, and the action is either "query" or "dynamicmap".

The detailed methods which require different permissions are listed in the following table:

methods	action
`Session.createObjectQuery(String)`	query
`EntityManager.createQuery(String)`	query
`Session.getMap(String)`	dynamicmap

ServerMapPermission types

An ServerMapPermission represents permissions to an ObjectMap hosted in a server. The name of the permission is the full name of the ObjectGrid map name, and the action is either "replicate" or "dynamicIndex".

The detailed methods which require different ServerMapPermission are listed in the following table:

methods	action
`ClientReplicableMap.enableClientReplication(Mode, int[], ReplicationMapListener)`	replicate
`BackingMap.createDynamicIndex(String, boolean, String, DynamicIndexCallback)`	dynamicIndex
`BackingMap.removeDynamicIndex(String)`	dynamicIndex

SecurityConstants

SecurityConstants class contains constants used for representing the security parameters.

---

com.ibm.websphere.objectgrid.security

Class SecurityConstants

java.lang.Object

↳ **com.ibm.websphere.objectgrid.security.SecurityConstants**

---

public class **SecurityConstants**
extends Object

This class contains the constants used for security configuration.

Since:

WAS XD 6.0, XC10

Field Summary

static int	**ACCESS_BY_CREATOR_ONLY_COMPLEMENT** The access by creator only authorization is enabled to complement the ObjectGrid map authorization.
static int	**ACCESS_BY_CREATOR_ONLY_DISABLED** The access by creator only authorization is disabled.
static int	**ACCESS_BY_CREATOR_ONLY_SUPERSEDE** The access by creator only authorization is enabled to supersede the ObjectGrid map authorization.
static int	**AUTHORIZATION_MECHANISM_CUSTOM** Constant representing custom authorization
st	

static int	**AUTHORIZATION_MECHANISM_JAAS** Constant representing JAAS authorization
static int	**CLIENT_CERTIFICATE_AUTHENTICATION_NEVER** Constant indicating client certificate authentication is not supported.
static int	**CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED** Constant indicating client certificate authentication is required.
static int	**CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED** Constant indicating client certificate authentication is supported.
static int	**CREDENTIAL_AUTHENTICATION_NEVER** Constant indicating credential authentication is not supported.
static int	**CREDENTIAL_AUTHENTICATION_REQUIRED** Constant indicating credential authentication is required.
static int	**CREDENTIAL_AUTHENTICATION_SUPPORTED** Constant indicating credential authentication is supported.
static	**NEVER_STRING**

static String	String representation for value Never indicating an option is not supported.
static String	**NEW_SECURE_TOKEN_MANAGER_STRING** String representation for "autoSecret" type of the secure token manager.
static String	**REQUIRED_STRING** String representation for value Required indicating an option is required.
static String	**SECURE_TOKEN_MANAGER_CUSTOM_STRING** String representation for "custom" type of the secure token manager.
static String	**SECURE_TOKEN_MANAGER_DEFAULT_STRING** String representation for "default" type of the secure token manager.
static String	**SECURE_TOKEN_MANAGER_NONE_STRING** String representation for "none" type of the secure token manager.
static int	**SSL_REQUIRED** Constant indicating SSL transport is required.

int	
static String	**SSL_REQUIRED_STRING** String representation for value SSL-Required indicating SSL transport type is required.
static int	**SSL_SUPPORTED** Constant indicating SSL transport is supported.
static String	**SSL_SUPPORTED_STRING** String representation for value SSL-Supported indicating SSL transport type is supported.
static String	**SUPPORTED_STRING** String representation for value Supported indicating an option is supported.
static int	**TCP_IP** Constant indicating TCP/IP is the only supported transport.
static String	**TCPIP_STRING** String representation for value TCP/IP indicating a transport type of TCP/IP is used.

Constructor Summary

SecurityConstants()

Method Summary

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

AUTHORIZATION_MECHANISM_JAAS

public static final int **AUTHORIZATION_MECHANISM_JAAS**

Constant representing JAAS authorization

See Also:
ObjectGrid.setAuthorizationMechanism(int), Constant Field Values

---

AUTHORIZATION_MECHANISM_CUSTOM

public static final int **AUTHORIZATION_MECHANISM_CUSTOM**

Constant representing custom authorization

See Also:
ObjectGrid.setAuthorizationMechanism(int), Constant Field Values

---

TCP_IP

public static final int **TCP_IP**

Constant indicating TCP/IP is the only supported transport.

If the client's transport type is set to this value, TCP/IP is the only supported transport type. If the server requires SSL, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1
See Also:
ClientSecurityConfiguration.setTransportType(int), Constant Field Values

---

SSL_SUPPORTED

public static final int **SSL_SUPPORTED**

Constant indicating SSL transport is supported.

If the client's transport type is set to this value, the client supports both TCP/IP and SSL. SSL will be used if both sides side supports SSL. Otherwise, TCP/IP will be used.

Since:
WAS XD 6.0.1
See Also:
ClientSecurityConfiguration.setTransportType(int), Constant Field Values

SSL_REQUIRED

`public static final int` **`SSL_REQUIRED`**

Constant indicating SSL transport is required.

If the client's transport type is set to this value, SSL is the only supported transport type. If the server requires TCP/IP, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setTransportType(int)`, Constant Field Values

CREDENTIAL_AUTHENTICATION_NEVER

`public static final int` **`CREDENTIAL_AUTHENTICATION_NEVER`**

Constant indicating credential authentication is not supported.

If the credential authentication type is set to this value, no credential authentication will be enforced. If the server requires credential authentication, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setCredentialAuthenticationType(int)`, Constant Field Values

CREDENTIAL_AUTHENTICATION_SUPPORTED

`public static final int` **`CREDENTIAL_AUTHENTICATION_SUPPORTED`**

Constant indicating credential authentication is supported.

If the credential authentication type is set to this value, credential authentication will be enforced if and only if both client and server support credential authentication.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setCredentialAuthenticationType(int)`, Constant Field Values

CREDENTIAL_AUTHENTICATION_REQUIRED

`public static final int` **`CREDENTIAL_AUTHENTICATION_REQUIRED`**

Constant indicating credential authentication is required.

If the credential authentication type is set to this value, credential authentication will be enforced. If the server doesn't support credential authentication, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setCredentialAuthenticationType(int)`, Constant Field Values

CLIENT_CERTIFICATE_AUTHENTICATION_NEVER

`public static final int` **`CLIENT_CERTIFICATE_AUTHENTICATION_NEVER`**

Constant indicating client certificate authentication is not supported.

If the client certificate authentication type is set to this value, no client certificate authentication will be enforced. If the server doesn't support client certificate authentication, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setClientCertificateAuthentication(int)`, Constant Field Values

CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED

`public static final int` **`CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED`**

Constant indicating client certificate authentication is supported.

If the client certificate authentication type is set to this value, client certificate authentication will be enforced when the following conditions are met:

- both client and server supports or requires client certificate authentication;
- the transport protocol to use is SSL;
- no credential authentication will be done.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setClientCertificateAuthentication(int)`, Constant Field Values

CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED

`public static final int` **`CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED`**

Constant indicating client certificate authentication is required. If the client certificate authentication type is set to this value, client certificate authentication will be enforced if the following conditions are met:

- both client and server supports or requires client certificate authentication;
- the transport protocol to use is SSL;
- no credential authentication will be done.

If the server doesn't support client certificate authentication and no credential authentication will be done, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setClientCertificateAuthentication(int)`, Constant Field Values

NEVER_STRING

`public static final String` **`NEVER_STRING`**

String representation for value Never indicating an option is not supported.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration settings "clientCertificateAuthentication" and "credentialAuthentication".

Since:
WAS XD 6.0.1

See Also:
`CLIENT_CERTIFICATE_AUTHENTICATION_NEVER`, `CREDENTIAL_AUTHENTICATION_NEVER`, Constant Field Values

SUPPORTED_STRING

```
public static final String SUPPORTED_STRING
```

String representation for value Supported indicating an option is supported.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration settings "clientCertificateAuthentication" and "credentialAuthentication".

Since:
WAS XD 6.0.1

See Also:
`CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED`, `CREDENTIAL_AUTHENTICATION_SUPPORTED`, Constant Field Values

REQUIRED_STRING

```
public static final String REQUIRED_STRING
```

String representation for value Required indicating an option is required.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration settings "clientCertificateAuthentication" and "credentialAuthentication".

Since:
WAS XD 6.0.1

See Also:
`CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED`, `CREDENTIAL_AUTHENTICATION_REQUIRED`, Constant Field Values

TCPIP_STRING

```
public static final String TCPIP_STRING
```

String representation for value TCP/IP indicating a transport type of TCP/IP is used.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration setting "transportType".

Since:
WAS XD 6.0.1

See Also:
`TCP_IP`, Constant Field Values

SSL_SUPPORTED_STRING

`public static final String` **`SSL_SUPPORTED_STRING`**

String representation for value SSL-Supported indicating SSL transport type is supported.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration setting "transportType".

Since:
WAS XD 6.0.1
See Also:
`SSL_SUPPORTED`, Constant Field Values

SSL_REQUIRED_STRING

`public static final String` **`SSL_REQUIRED_STRING`**

String representation for value SSL-Required indicating SSL transport type is required.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration setting "transportType".

Since:
WAS XD 6.0.1
See Also:
`SSL_REQUIRED`, Constant Field Values

SECURE_TOKEN_MANAGER_NONE_STRING

`public static final String` **`SECURE_TOKEN_MANAGER_NONE_STRING`**

String representation for "none" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType".

Since:
WAS XD 6.0.1
See Also:
Constant Field Values

SECURE_TOKEN_MANAGER_DEFAULT_STRING

`public static final String` **`SECURE_TOKEN_MANAGER_DEFAULT_STRING`**

String representation for "default" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType". This value requires users to provide the secure token key store settings.

Since:
WAS XD 6.0.1
See Also:
Constant Field Values

SECURE_TOKEN_MANAGER_CUSTOM_STRING

```
public static final String SECURE_TOKEN_MANAGER_CUSTOM_STRING
```

String representation for "custom" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType". This value requires users to provide the SecureTokenManager implementation class name using the "customSecureTokenManagerClass" configuration setting.

Since:
WAS XD 6.0.1
See Also:
Constant Field Values

NEW_SECURE_TOKEN_MANAGER_STRING

```
public static final String NEW_SECURE_TOKEN_MANAGER_STRING
```

String representation for "autoSecret" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType". This value does not require users to provide other settings.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

ACCESS_BY_CREATOR_ONLY_DISABLED

```
public static final int ACCESS_BY_CREATOR_ONLY_DISABLED
```

The access by creator only authorization is disabled.

The access by creator authorization ensures that only the user (represented by the Principals associated with it), who inserts the data entry into the map, can access the data. Here the access means read, update, invalidate, and remove.

Since:
WAS XD 6.1 FIX3
See Also:
Constant Field Values

ACCESS_BY_CREATOR_ONLY_COMPLEMENT

```
public static final int ACCESS_BY_CREATOR_ONLY_COMPLEMENT
```

The access by creator only authorization is enabled to complement the ObjectGrid map authorization.

The access by creator authorization ensures that only the user (represented by the Principals associated with it), who inserts the data entry into the map, can access the data. Here the access means read, update, invalidate, and remove.

If this constant is used, both map authorization and access by creator only authorization will take effect. Therefore, you can further limit the operations to the data entries. For example, you can restrict the creator from invalidating the data entries.

Since:
WAS XD 6.1 FIX3
See Also:
Constant Field Values

ACCESS_BY_CREATOR_ONLY_SUPERSEDE

`public static final int` **`ACCESS_BY_CREATOR_ONLY_SUPERSEDE`**

The access by creator only authorization is enabled to supersede the ObjectGrid map authorization.

The access by creator authorization ensures that only the user (represented by the Principals associated with it), who inserts the data entry into the map, can access the data. Here the access means read, update, invalidate, and remove.

If this constant is used, the access by creator only authorization will supersede the map authorization; no map authorization will be done.

Since:
WAS XD 6.1 FIX3
See Also:
Constant Field Values

Constructor Detail

SecurityConstants

`public` **`SecurityConstants`**`()`

com.ibm.websphere.objectgrid.security

Class ObjectGridSecurityException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.security.ObjectGridSecurityException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

CannotGenerateCredentialException, ExpiredCredentialException, InvalidCredentialException

---

```
public class ObjectGridSecurityException
extends ObjectGridException
```

This exception represents a general ObjectGrid security exception.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridSecurityException() Constructs a new ObjectGridSecurityException with `null` as its detail message.
ObjectGridSecurityException(String message) Constructs a new ObjectGridSecurityException with the specified detail message.
ObjectGridSecurityException(String message, Throwable cause) Constructs a new ObjectGridSecurityException with the specified detail message and cause.
ObjectGridSecurityException(Throwable cause) Constructs a new ObjectGridSecurityException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridSecurityException

public **ObjectGridSecurityException**()

Constructs a new ObjectGridSecurityException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

ObjectGridSecurityException

public **ObjectGridSecurityException**(String message)

Constructs a new ObjectGridSecurityException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
ObjectGridException.initCause(Throwable), Throwable.getMessage()

ObjectGridSecurityException

public **ObjectGridSecurityException**(String message,
Throwable cause)

Constructs a new ObjectGridSecurityException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridSecurityException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
ObjectGridException.getCause(), Throwable.getMessage()

ObjectGridSecurityException

public **ObjectGridSecurityException**(Throwable cause)

Constructs a new ObjectGridSecurityException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for

ObjectGridSecurityExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

Package com.ibm.websphere.objectgrid.spring

This package holds the Spring specific APIs for ObjectGrid.

See:

Description

Interface Summary

SpringLocalTxManager	This interface has the methods for interacting with the ObjectGrid Spring LocalTransactionManager.

Class Summary

ObjectGridCache	This class is a WebSphere eXtreme Scale implementation of the Spring Framework's Cache interface.
ObjectGridCatalogServiceDomainBean	This class is a Spring bean representing a Catalog Service Domain for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.
ObjectGridClientBean	This class is a Spring bean representing an ObjectGrid client instance for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.
ObjectGridSpringFactory	This class serves as a factory to construct instances of the various Spring specific APIs.

Exception Summary

CannotGetObjectGridSessionException	This can be thrown by getSession if a new Session cannot be obtained for any reason.
ObjectGridTransactionException	The ObjectGrid platform transaction manager can throw this unchecked exception whenever errors occur.

Package com.ibm.websphere.objectgrid.spring Description

This package holds the Spring specific APIs for ObjectGrid.

Local Transaction Support

ObjectGrid has implemented a Spring PlatformTransactionManager. This allows Spring to manage local transactions using a single ObjectGrid session. Spring can then be used to annotate POJOs with container managed transaction semantics much like a J2EE application server does using J2EE CMT. An application should instantiate a SpringLocalTxManager using the appropriate factory method on ObjectGridSpringFactory and then wire a reference to that object in to all POJOs that use Spring CMT. This instance has a getSession method to obtain the correct Session for that POJO. The application must call one of the SpringLocalTxManager#setObjectGridForThread methods before invoking a managed POJO to specify which ObjectGrid instance should be used for any CMT on this thread.

---

com.ibm.websphere.objectgrid.spring

Interface SpringLocalTxManager

public interface **SpringLocalTxManager**

This interface has the methods for interacting with the ObjectGrid Spring LocalTransactionManager. It also allows the desired partition to use with this thread to be specified.

Since:
WAS XD 6.1 FIX3, XC10

See Also:
ObjectGridSpringFactory.getLocalPlatformTransactionManager()

Method Summary

Session	**getSession**() This returns a managed session for the ObjectGrid associated with this thread.
void	**setObjectGridForThread**(ObjectGrid grid) This indicates the ObjectGrid to use on this thread when a session is requested.

Method Detail

setObjectGridForThread

void **setObjectGridForThread**(ObjectGrid grid)

This indicates the ObjectGrid to use on this thread when a session is requested. This replaces any previously associated ObjectGrid, i.e. only a single grid instance can be associated with a thread at a time.

Parameters:
grid - the ObjectGrid to set on this thread.

See Also:
getSession()

getSession

Session **getSession**()

This returns a managed session for the ObjectGrid associated with this thread.

Do not call `begin()`, `commit()` or `rollback()` directly on the session. Spring manages the transaction automatically.

Returns:

A managed Session to use with this thread.

Throws:

`CannotGetObjectGridSessionException` - thrown when an ObjectGrid session can't be retrieved.

See Also:

`setObjectGridForThread(ObjectGrid)`

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridTransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        org.springframework.core.NestedRuntimeException
          org.springframework.transaction.TransactionException
            com.ibm.websphere.objectgrid.spring.ObjectGridTransactionException

All Implemented Interfaces:

Serializable

---

public class **ObjectGridTransactionException**
extends TransactionException

The ObjectGrid platform transaction manager can throw this unchecked exception whenever errors occur.

Since:

WAS XD 6.1 FIX3, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridTransactionException(String message) Constructs a new ObjectGridTransactionException with the specified detail message.
ObjectGridTransactionException(String message, Throwable cause) Constructs a new ObjectGridTransactionException with the specified detail message and cause.

Method Summary

Methods inherited from class org.springframework.core.NestedRuntimeException
contains, getMessage, getMostSpecificCause, getRootCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridTransactionException

```
public ObjectGridTransactionException(String message,
                                      Throwable cause)
```

Constructs a new ObjectGridTransactionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridTransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`Throwable.getCause()`, `NestedRuntimeException.getMessage()`

ObjectGridTransactionException

```
public ObjectGridTransactionException(String message)
```

Constructs a new ObjectGridTransactionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`Throwable.initCause(Throwable)`, `NestedRuntimeException.getMessage()`

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridSpringFactory

java.lang.Object

↳ **com.ibm.websphere.objectgrid.spring.ObjectGridSpringFactory**

```
public final class ObjectGridSpringFactory
extends Object
```

This class serves as a factory to construct instances of the various Spring specific APIs.

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static String	**SCOPE_SHARD** Scope identifier for shard scope: "shard".

Constructor Summary

ObjectGridSpringFactory()

Method Summary

static BeanFactory	**getBeanFactoryForObjectGrid**(String objectGridName) This returns the currently registered external bean factory for a named object grid.
sta	

static Object	**getBeanInShardScope**(ObjectGrid shard, String beanName) This returns an instance of the named Spring bean with the current shard scope using the specified ObjectGrid instance.
static SpringLocalTxManager	**getLocalPlatformTransactionManager**() This returns an ObjectGrid PlatformLocalTransactionManager.
static void	**registerSpringBeanFactoryAdapter**(String objectGridName, Object springBeanFactory) This returns an adapter for a Spring based bean factory.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SCOPE_SHARD

public static final String **SCOPE_SHARD**

Scope identifier for shard scope: "shard".

See Also:

getBeanInShardScope(ObjectGrid, String), Constant Field Values

Constructor Detail

ObjectGridSpringFactory

public **ObjectGridSpringFactory**()

Method Detail

getLocalPlatformTransactionManager

```
public static SpringLocalTxManager getLocalPlatformTransactionManager()
```

This returns an ObjectGrid PlatformLocalTransactionManager.

Returns:
the PlatformLocalTransactionManager instance.

---

registerSpringBeanFactoryAdapter

```
public static void registerSpringBeanFactoryAdapter(String objectGridName,
                                                    Object springBeanFactory)
                                             throws ClassCastException
```

This returns an adapter for a Spring based bean factory. We use an Object type here to avoid making ObjectGrid dependent on Spring classes being present. A ClassCastException exception is thrown if the supplied factory isn't a Spring BeanFactory instance.

Parameters:
`objectGridName` - the name of the ObjectGrid
`springBeanFactory` - A Spring BeanFactory instance.
Throws:
`ClassCastException` - thrown when the Object type is not a BeanFactory instance.

---

getBeanFactoryForObjectGrid

```
public static BeanFactory getBeanFactoryForObjectGrid(String objectGridName)
```

This returns the currently registered external bean factory for a named object grid. If no factory has been registered then it attempts to construct a Spring BeanFactory using the xml resource on the class path @ "/X_spring.xml" and /META-INF/X_spring.xml where X is the name of the ObjectGrid. If the xml file is on the class path then the ObjectGrid name MUST be a valid resource name.

Parameters:
`objectGridName` - The name of the ObjectGrid
Returns:
The BeanFactory instance or null if there were none registered

---

getBeanInShardScope

```
public static Object getBeanInShardScope(ObjectGrid shard,
                                         String beanName)
```

This returns an instance of the named Spring bean with the current shard scope using the specified ObjectGrid instance. This allows shard scoped beans to be obtained.

Parameters:
`shard` - The ObjectGrid instance to use to scope Spring beans using "shard" as scope.
`beanName` - The bean to return
Returns:
The bean instance if it exists
See Also:
`SCOPE_SHARD`

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

com.ibm.websphere.objectgrid.spring

Class ObjectGridClientBean

java.lang.Object
  com.ibm.websphere.objectgrid.spring.ObjectGridClientBean

All Implemented Interfaces:

InitializingBean

```
public final class ObjectGridClientBean
extends Object
implements InitializingBean
```

This class is a Spring bean representing an ObjectGrid client instance for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.

Users must provide a ObjectGridCatalogServiceDomainBean to configure this client. The ObjectGrid name is optional when using the provided XML configuration files.

Since:

8.5, XC10

Constructor Summary

ObjectGridClientBean()

Method Summary

void	**afterPropertiesSet**() Initializes the client bean.
void	**setCatalogServiceDomain**(ObjectGridCatalogServiceDomainBean catalogServiceDomain) Sets the ObjectGridCatalogServiceDomainBean used by the client.
void	**setObjectGridName**(String objectGridName) Sets the name of the ObjectGrid for the client.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridClientBean

`public` **`ObjectGridClientBean`**`()`

Method Detail

setObjectGridName

`public void` **`setObjectGridName`**`(`String `objectGridName)`

Sets the name of the ObjectGrid for the client. This is optional when using the provided XML configuration files.

Parameters:

`objectGridName` - The name of the ObjectGrid to connect to.

---

setCatalogServiceDomain

`public void` **`setCatalogServiceDomain`**`(`ObjectGridCatalogServiceDomainBean `catalogServiceDomain)`

Sets the ObjectGridCatalogServiceDomainBean used by the client.

Parameters:

`catalogServiceDomain` - The ObjectGridCatalogServiceDomainBean for the client.

---

afterPropertiesSet

`public void` **`afterPropertiesSet`**`()`
`throws` Exception

Initializes the client bean. This method is to be called after all setter methods have been called and will throw an IllegalArgumentException if setCatalogServiceDomain(ObjectGridCatalogServiceDomainBean) has not been called prior.

Specified by:

afterPropertiesSet in interface InitializingBean

Throws:

Exception - if setCatalogServiceDomain(ObjectGridCatalogServiceDomainBean) has not been called prior or an error occurs retrieving the ObjectGrid.

See Also:

InitializingBean.afterPropertiesSet()

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridCatalogServiceDomainBean

java.lang.Object
  com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean

All Implemented Interfaces:

InitializingBean

```
public final class ObjectGridCatalogServiceDomainBean
extends Object
implements InitializingBean
```

This class is a Spring bean representing a Catalog Service Domain for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.

Users must provide the catalog service endpoints used to connect to the eXtreme Scale cluster/domain. Users optionally may provide a client override XML and/or a client security configuration.

Since:

8.5, XC10

Constructor Summary

ObjectGridCatalogServiceDomainBean()

Method Summary

void	**afterPropertiesSet**() Initializes the connection to the Catalog Service Domain.
void	**setCatalogServiceEndpoints**(String catalogServiceEndpoints) Sets the catalog service endpoints used to connect to the cluster/domain.
void	**setClientOverrideXml**(Resource clientOverrideXml) Sets the location of the client override XML.
void	**setClientSecurityConfig**(Resource clientSecurityConfiguration) Sets the location of the client security configuration.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridCatalogServiceDomainBean

public **ObjectGridCatalogServiceDomainBean**()

Method Detail

setCatalogServiceEndpoints

public void **setCatalogServiceEndpoints**(String catalogServiceEndpoints)

Sets the catalog service endpoints used to connect to the cluster/domain.

Parameters:

catalogServiceEndpoints - The catalog service endpoints to connect to the cluster/domain

---

setClientOverrideXml

public void **setClientOverrideXml**(Resource clientOverrideXml)

Sets the location of the client override XML.

Parameters:

clientOverrideXml - The location of the client override XML.

---

setClientSecurityConfig

public void **setClientSecurityConfig**(Resource clientSecurityConfiguration)

Sets the location of the client security configuration.

Parameters:

clientSecurityConfiguration - The location of the client security configuration.

---

afterPropertiesSet

public void **afterPropertiesSet**()
throws Exception

Initializes the connection to the Catalog Service Domain. This method is to be called after all setter methods have been called and will throw an IllegalArgumentException if setCatalogServiceEndpoints(String) have not been called prior.

Specified by:

afterPropertiesSet in interface InitializingBean

Throws:

Exception - if setCatalogServiceEndpoints(String) have not been called prior or an error occurs initializing the connection.

See Also:

InitializingBean.afterPropertiesSet()

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridCache

java.lang.Object
  com.ibm.websphere.objectgrid.spring.ObjectGridCache

All Implemented Interfaces:
InitializingBean, Cache

```
public final class ObjectGridCache
extends Object
implements Cache, InitializingBean
```

This class is a WebSphere eXtreme Scale implementation of the Spring Framework's Cache interface.

Users must provide a name and a ObjectGridClientBean to configure this cache. The ObjectMap name is optional when using the provided XML configuration files.

This implementation allows for the storage of null values and does not support null keys.

The following Spring Inversion of Control (IoC) container configuration snippet creates two caches, named default and books hosted by the catalog service domain with connection end-points of host1:2809,host2:2809.

```
 <bean id="wxsCSDomain" class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainB
ean"
     p:catalog-service-endpoints="host1:2809,host2:2809" />

 <bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
      p:catalog-service-domain-ref="wxsCSDomain" />

 <bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
     <property name="caches">
         <set>
             <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
                 p:name="default"
                 p:object-grid-client-ref="wxsGridClient" />
             <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
                 p:name="books"
                 p:object-grid-client-ref="wxsGridClient" />
         </set>
     </property>
 </bean>
```

Since:
8.5, XC10

Nested Class Summary

Nested classes/interfaces inherited from interface

org.springframework.cache.Cache
Cache.ValueWrapper

Constructor Summary

ObjectGridCache()

Method Summary

void	**afterPropertiesSet**() Initializes this cache.
void	**clear**() Clears all entries from the cache.
void	**evict**(Object key) Evicts the entry at the given key.
Cache.ValueWrapper	**get**(Object key) Retrieves the object from the cache at the given key.
String	**getName**() Returns the name of the cache.
Object	**getNativeCache**() This implementation returns `null`.
void	**put**(Object key, Object value) Creates an entry in the cache.
void	**setMapName**(String mapName) Sets the ObjectMap name.

void	**setName**(String name) Sets the cache name.
void	**setObjectGridClient**(ObjectGridClientBean objectGridClient) Sets the ObjectGridClientBean to use.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridCache

public **ObjectGridCache**()

Method Detail

afterPropertiesSet

public void **afterPropertiesSet**()
throws Exception

Initializes this cache. This method is to be called after all setter methods have been called and will throw an IllegalArgumentException if setObjectGridClient(ObjectGridClientBean) or setName(String) have not been called prior.

Specified by:
afterPropertiesSet in interface InitializingBean

Throws:
Exception - if setObjectGridClient(ObjectGridClientBean) or setName(String) have not been called prior or an error occurs initializing the cache.

See Also:
InitializingBean.afterPropertiesSet()

---

getName

public String **getName**()

Returns the name of the cache.

Specified by:
getName in interface Cache

Returns:
The name of the cache.

See Also:
Cache.getName()

---

setName

public void **setName**(String name)

Sets the cache name.

Parameters:

`name` - The cache name.

See Also:

`getName()`

---

setMapName

```
public void setMapName(String mapName)
```

Sets the ObjectMap name. This is optional when using the provided XML configuration files.

Parameters:

`mapName` - The name of the ObjectMap

---

setObjectGridClient

```
public void setObjectGridClient(ObjectGridClientBean objectGridClient)
```

Sets the `ObjectGridClientBean` to use.

Parameters:

`objectGridClient` - The `ObjectGridClientBean` to use

---

getNativeCache

```
public Object getNativeCache()
```

This implementation returns `null`.

Specified by:

`getNativeCache` in interface `Cache`

Returns:

This implementation returns `null`.

See Also:

`Cache.getNativeCache()`

---

get

```
public Cache.ValueWrapper get(Object key)
```

Retrieves the object from the cache at the given key. Returns `null` if there is no mapping for the key or if the key is `null`

Specified by:

`get` in interface `Cache`

Returns:

the object from the cache at the given key or `null` if there is no mapping or the key is `null`

See Also:

`Cache.get(java.lang.Object)`

---

put

```
public void put(Object key,
                Object value)
```

Creates an entry in the cache. Overwrites the value if an entry for the given key exists. This method does not create an entry if the given key is `null`.

Specified by:
`put` in interface `Cache`

See Also:
`Cache.put(java.lang.Object, java.lang.Object)`

---

evict

`public void **evict**(Object key)`

Evicts the entry at the given key. If the provided key is `null` this method does not evict any entries.

Specified by:
`evict` in interface `Cache`

See Also:
`Cache.evict(java.lang.Object)`

---

clear

`public void **clear**()`

Clears all entries from the cache.

Specified by:
`clear` in interface `Cache`

See Also:
`Cache.clear()`

---

com.ibm.websphere.objectgrid.spring

Class CannotGetObjectGridSessionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        org.springframework.core.NestedRuntimeException
          com.ibm.websphere.objectgrid.spring.CannotGetObjectGridSessionException

All Implemented Interfaces:

Serializable

```
public class CannotGetObjectGridSessionException
extends NestedRuntimeException
```

This can be thrown by getSession if a new Session cannot be obtained for any reason.

Since:

WAS XD 6.1 FIX3, XC10

See Also:

Serialized Form

Constructor Summary

CannotGetObjectGridSessionException(String message) Constructs a new CannotGetObjectGridSessionException with the specified detail message.
CannotGetObjectGridSessionException(String message, Throwable cause) Constructs a new CannotGetObjectGridSessionException with the specified detail message and cause.

Method Summary

Methods inherited from class org.springframework.core.NestedRuntimeException
contains, getMessage, getMostSpecificCause, getRootCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CannotGetObjectGridSessionException

```
public CannotGetObjectGridSessionException(String message,
                                           Throwable cause)
```

Constructs a new CannotGetObjectGridSessionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this CannotGetObjectGridSessionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`Throwable.getCause()`, `NestedRuntimeException.getMessage()`

CannotGetObjectGridSessionException

```
public CannotGetObjectGridSessionException(String message)
```

Constructs a new CannotGetObjectGridSessionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`Throwable.initCause(Throwable)`, `NestedRuntimeException.getMessage()`

---

com.ibm.websphere.objectgrid

Class UndefinedMapException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.UndefinedMapException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class UndefinedMapException
extends ObjectGridException
```

This exception indicates that the map which an application tries to access is not defined in the ObjectGrid.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

UndefinedMapException() Constructs a new UndefinedMapException with `null` as its detail message.
UndefinedMapException(String message) Constructs a new UndefinedMapException with the specified detail message.
UndefinedMapException(String message, Throwable cause) Constructs a new UndefinedMapException with the specified detail message and cause.
UndefinedMapException(Throwable cause) Constructs a new UndefinedMapException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

UndefinedMapException

public **UndefinedMapException**()

Constructs a new UndefinedMapException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

UndefinedMapException

public **UndefinedMapException**(String message)

Constructs a new UndefinedMapException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

UndefinedMapException

public **UndefinedMapException**(String message,
Throwable cause)

Constructs a new UndefinedMapException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this UndefinedMapException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

UndefinedMapException

public **UndefinedMapException**(Throwable cause)

Constructs a new UndefinedMapException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for UndefinedMapExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that

the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class UnavailableServiceException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException
          com.ibm.websphere.objectgrid.UnavailableServiceException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

TransactionAffinityException, TransactionQuiesceException

```
public class UnavailableServiceException
extends LoaderException
```

This exception is thrown when all servers are dead or when all services are unavailable even though servers are running.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

UnavailableServiceException() Constructs a new UnavailableServiceException with `null` as its detail message.
UnavailableServiceException(String message) Constructs a new UnavailableServiceException with the specified detail message.
UnavailableServiceException(String message, Throwable cause) Constructs a new UnavailableServiceException with the specified detail message and cause.
UnavailableServiceException(Throwable cause) Constructs a new UnavailableServiceException with a specified cause.

Method Summary

int	**getReplicationGroup**() Returns the replication group identifier for this exception.
void	**setReplicationGroup**(int replicationGroup) Sets the replication group identifier for this exception.

d	

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

UnavailableServiceException

```
public UnavailableServiceException()
```

Constructs a new UnavailableServiceException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

UnavailableServiceException

```
public UnavailableServiceException(String message)
```

Constructs a new UnavailableServiceException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

UnavailableServiceException

```
public UnavailableServiceException(String message,
                                   Throwable cause)
```

Constructs a new UnavailableServiceException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this UnavailableServiceException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

UnavailableServiceException

`public` **`UnavailableServiceException`**`(`[`Throwable`]` cause)`

Constructs a new UnavailableServiceException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for UnavailableServiceExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

[`ObjectGridException.getCause()`]

Method Detail

getReplicationGroup

`public int` **`getReplicationGroup`**`()`

Returns the replication group identifier for this exception.

Returns:

the argument that was passed to the `setReplicationGroup(int)` method of this class or `0` if the `setReplicationGroup` method was not previously called for this object.

See Also:

[`setReplicationGroup(int)`]

setReplicationGroup

`public void` **`setReplicationGroup`**`(int replicationGroup)`

Sets the replication group identifier for this exception.

Parameters:

`replicationGroup` - The replication group identifier

com.ibm.websphere.objectgrid

Interface TxID

All Superinterfaces:

Serializable

```
public interface TxID
extends Serializable
```

This interface is an opaque identifier for a transaction. Context information can be stored and retrieved in multiple slots on this object. This mechanism allows a `TransactionCallback` and `Loader`, for example, to share state information with each other in the context of a specific session transaction.

The TxID.toString() output can be used to determine whether the originating Session transaction is a single partition transaction or a multi-partition transaction. If the String output begins with the keyword `Local` then this indicates a single partition transaction, for example: `Local-40000139-72B2-C037-E000-1C271366B073`

If the String output begins with the keyword `WXS` then this indicates a multi-partition transaction, for example: `WXS-40000139-72B2-BD3A-E000-1C271366B073`

Since:

WAS XD 6.0, XC10

See Also:

`Loader`, `ObjectGrid.reserveSlot(String)`, `Session`, `TransactionCallback`

Field Summary

static String	**SLOT_NAME** All slots should be reserved using this name.

Method Summary

boolean	**equals**(TxID o) Checks for equality between two TxID objects.
Se	

Session	**getSession**() Returns the Session that owns this TxID.
Object	**getSlot**(int slotNumber) Gets the context information currently associated with this transaction.
int	**hashCode**() Returns the hashcode of the Tx identifier.
void	**putSlot**(int slotNumber, Object o) Sets some context information to be associated with this transaction.

Field Detail

SLOT_NAME

static final String **SLOT_NAME**

All slots should be reserved using this name.

See Also:
ObjectGrid.reserveSlot(String), Constant Field Values

Method Detail

equals

boolean **equals**(TxID o)

Checks for equality between two TxID objects.

Parameters:
o - Input TxID to check for equality against
Returns:
true, if they are equal; false, if they not equal

---

hashCode

int **hashCode**()

Returns the hashcode of the Tx identifier.

Overrides:
hashCode in class Object

Returns:
hashcode

---

getSlot

`Object` **`getSlot`**`(int slotNumber)`

Gets the context information currently associated with this transaction.

Parameters:
`slotNumber` - the slot number for the context information being requested
Returns:
Object the current context information for the slot number
See Also:
`putSlot(int, Object)`, `ObjectGrid.reserveSlot(String)`

---

putSlot

`void` **`putSlot`**`(int slotNumber,`
`Object o)`

Sets some context information to be associated with this transaction.

Parameters:
`slotNumber` - the slot number
`o` - Object to be put into the TxID slot
See Also:
`getSlot(int)`, `ObjectGrid.reserveSlot(String)`

---

getSession

`Session` **`getSession`**`()`

Returns the Session that owns this TxID.

Returns:
a Session object to use.
See Also:
`Session`

---

com.ibm.websphere.objectgrid

Class TransactionTimeoutException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        com.ibm.websphere.objectgrid.ObjectGridRuntimeException
          com.ibm.websphere.objectgrid.TransactionTimeoutException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionTimeoutException**
extends ObjectGridRuntimeException

This exception is thrown when a transaction exceeds the transaction timeout that was specified on the `ObjectGrid` or `Session`.

Since:

WAS XD 6.0.1, XC10

See Also:

ObjectGrid.setTxTimeout(int), Session.setTransactionTimeout(int), Serialized Form

---

Constructor Summary

TransactionTimeoutException(String message, String txIdString) Constructs a new TransactionTimeoutException with the specified detail message.
TransactionTimeoutException(String message, Throwable cause)

Method Summary

String	**getTxIDString**() Get the String representation of the `TxID` for the transaction that timed out.
Date	**whenOccurred**() Gives the time when this TransactionTimeoutException was created.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridRuntimeException

getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionTimeoutException

```
public TransactionTimeoutException(String message,
                                   Throwable cause)
```

TransactionTimeoutException

```
public TransactionTimeoutException(String message,
                                   String txIdString)
```

Constructs a new TransactionTimeoutException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`txIdString` - the result of `TxID.toString()` for the transaction that timed out.

See Also:

ObjectGridRuntimeException.initCause(Throwable), Throwable.getMessage()

Method Detail

whenOccurred

```
public Date whenOccurred()
```

Gives the time when this TransactionTimeoutException was created.

Returns:

Date object that represents the instant in time when this exception object was created.

---

getTxIDString

```
public String getTxIDString()
```

Get the String representation of the `TxID` for the transaction that timed out.

Returns:

String value of TxID of transaction that timed out.

---

com.ibm.websphere.objectgrid

Class TransactionQuiesceException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException
          com.ibm.websphere.objectgrid.UnavailableServiceException
            com.ibm.websphere.objectgrid.TransactionQuiesceException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionQuiesceException**
extends UnavailableServiceException

This exception is thrown when partition/shard/mapset/replication group/ replication group member/server/cluster/objectgrid is entered quiesce process for various reasons such as shard movement, partition relocation, system update, server shutdown, and others. Quiesce process ensures data integrity and transaction integrity. This exception is thrown only for new start of a new transaction; it will not impact old transaction requests that are allowed to finish.

Since:

WAS XD 6.1, XC10

See Also:

Serialized Form

Constructor Summary

TransactionQuiesceException() Constructs a new TransactionQuiesceException with `null` as its detail message.
TransactionQuiesceException(String message) Constructs a new TransactionQuiesceException with the specified detail message.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.UnavailableServiceException

getReplicationGroup, setReplicationGroup

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionQuiesceException

public **TransactionQuiesceException**()

Constructs a new TransactionQuiesceException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

TransactionQuiesceException

public **TransactionQuiesceException**(String message)

Constructs a new TransactionQuiesceException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

---

com.ibm.websphere.objectgrid

Class TransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.TransactionException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

SessionNotReentrantException, TransactionAlreadyActiveException

---

```
public class TransactionException
extends ObjectGridException
```

A general transaction exception indicating something went wrong with a transaction. The `isTransactionActive()` and `wasTransactionRolledBack()` methods can be used to determine whether transaction is still active or was rolled back as a result of this exception occuring.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Field Summary

protected boolean	**ivTransactionRolledBack** Indicates whether the transaction was rolled back or not.

Constructor Summary

TransactionException(String message, boolean rolledBack) Constructs a new TransactionException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception.

TransactionException(String message, Throwable cause, boolean rolledBack) Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String message, TransactionException cause, boolean rolledBack) Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.
TransactionException(Throwable cause, boolean rolledBack) Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception.
TransactionException(TransactionException cause, boolean rolledBack) Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception.

Method Summary

boolean	**isTransactionActive**() Returns true if the transaction is active.
boolean	**wasTransactionRolledBack**() Returns true if the transaction was rolled back.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ivTransactionRolledBack

```
protected boolean ivTransactionRolledBack
```

Indicates whether the transaction was rolled back or not.

Constructor Detail

TransactionException

```
public TransactionException(String message,
                            boolean rolledBack)
```

Constructs a new TransactionException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(Throwable cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for as a wrapper for other Throwable objects that occur.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

`ObjectGridException.getCause()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(TransactionException cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for as a wrapper for other Throwable objects that occur.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

Since:

WAS XD 6.1 IFIX1

See Also:

`ObjectGridException.getCause()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(String message,
                            Throwable cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(String message,
                            TransactionException cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
`rolledBack` - A value of `true` indicates the transaction was rolled back.

Since:

WAS XD 6.1 IFIX1

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`, `wasTransactionRolledBack()`

Method Detail

isTransactionActive

```
public boolean isTransactionActive()
```

Returns `true` if the transaction is active. Otherwise, `false` is returned to indicate either the transaction never started or was completed.

Returns:

true if the transaction is active

---

wasTransactionRolledBack

```
public boolean wasTransactionRolledBack()
```

Returns `true` if the transaction was rolled back.

Returns:

true if the transaction was rolled back

---

com.ibm.websphere.objectgrid

Class TransactionAlreadyActiveException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.TransactionException
          com.ibm.websphere.objectgrid.TransactionAlreadyActiveException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionAlreadyActiveException**
extends TransactionException

An exception indicating a transaction is already active for the current session. This exception does not cause the current active transaction to be rolled back, so the `isTransactionActive` method will return `true`.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Field Summary

Fields inherited from class com.ibm.websphere.objectgrid.TransactionException
ivTransactionRolledBack

Constructor Summary

TransactionAlreadyActiveException() Constructs a new TransactionAlreadyActiveException with `null` as its detail message.
TransactionAlreadyActiveException(String message) Constructs a new TransactionAlreadyActiveException with the specified detail message.
TransactionAlreadyActiveException(String message, Throwable cause) Constructs a new TransactionAlreadyActiveException with the specified detail message and cause.
TransactionAlreadyActiveException(Throwable cause) Constructs a new TransactionAlreadyActiveException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.TransactionException
isTransactionActive, wasTransactionRolledBack

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionAlreadyActiveException

public **TransactionAlreadyActiveException**()

Constructs a new TransactionAlreadyActiveException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

TransactionAlreadyActiveException

public **TransactionAlreadyActiveException**(String message)

Constructs a new TransactionAlreadyActiveException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

TransactionAlreadyActiveException

public **TransactionAlreadyActiveException**(Throwable cause)

Constructs a new TransactionAlreadyActiveException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for TransactionAlreadyActiveExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:
ObjectGridException.getCause()

TransactionAlreadyActiveException

```
public TransactionAlreadyActiveException(String message,
                                         Throwable cause)
```

Constructs a new TransactionAlreadyActiveException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionAlreadyActiveException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
ObjectGridException.getCause(), Throwable.getMessage()

---

com.ibm.websphere.objectgrid

Class TransactionAffinityException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException
          com.ibm.websphere.objectgrid.UnavailableServiceException
            com.ibm.websphere.objectgrid.TransactionAffinityException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionAffinityException**
extends UnavailableServiceException

This exception is thrown for inflight transaction when server fails over. We suggest applications to retry transaction.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

TransactionAffinityException() Constructs a new TransactionAffinityException with `null` as its detail message.
TransactionAffinityException(String message) Constructs a new TransactionAffinityException with the specified detail message.
TransactionAffinityException(String message, Throwable cause) Constructs a new TransactionAffinityException with the specified detail message and cause.
TransactionAffinityException(Throwable cause) Constructs a new TransactionAffinityException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.UnavailableServiceException
getReplicationGroup, setReplicationGroup

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionAffinityException

`public` **`TransactionAffinityException`**`()`

Constructs a new TransactionAffinityException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

TransactionAffinityException

`public` **`TransactionAffinityException`**`(String message)`

Constructs a new TransactionAffinityException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

TransactionAffinityException

`public` **`TransactionAffinityException`**`(String message,`
`Throwable cause)`

Constructs a new TransactionAffinityException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionAffinityException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

TransactionAffinityException

`public` **`TransactionAffinityException`**`(Throwable cause)`

Constructs a new TransactionAffinityException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for TransactionAffinityExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class TargetNotAvailableException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        com.ibm.websphere.objectgrid.TargetNotAvailableException

All Implemented Interfaces:

Serializable

---

public class **TargetNotAvailableException**
extends RuntimeException

A `TargetNotAvailableException` indicates the ObjectGrid target is not available. This could be due to the fact that ObjectGrid servers are not available or the ObjectGrid placement has not finished.

Since:

WAS XD 6.1, XC10

See Also:

Serialized Form

Constructor Summary

TargetNotAvailableException() Constructs a new TargetNotAvailableException with `null` as its detail message.
TargetNotAvailableException(String message) Constructs a new TargetNotAvailableException with the specified detail message.
TargetNotAvailableException(String message, Throwable cause) Constructs a new TargetNotAvailableException with the specified detail message and cause.
TargetNotAvailableException(Throwable cause) Constructs a new TargetNotAvailableException with a specified cause.

Method Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

---

Constructor Detail

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`()`

Constructs a new TargetNotAvailableException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

`Throwable.initCause(Throwable)`

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`(String message)`

Constructs a new TargetNotAvailableException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`Throwable.initCause(Throwable)`, `Throwable.getMessage()`

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`(Throwable cause)`

Constructs a new TargetNotAvailableException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for TargetNotAvailableExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`Throwable.getCause()`

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`(String message,`
`Throwable cause)`

Constructs a new TargetNotAvailableException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TargetNotAvailableException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

Throwable.getCause(), Throwable.getMessage()

---

com.ibm.websphere.objectgrid

Class TTLType

java.lang.Object

  com.ibm.websphere.objectgrid.TTLType

All Implemented Interfaces:

Serializable

```
public class TTLType
extends Object
implements Serializable
```

Every BackingMap in ObjectGrid has a built in timed based evictor that is referred to as "time to live" evictor or TTL evictor. Each BackingMap entry has an expiration time that determines how long the entry is allowed to live in the BackingMap. When the expiration time is reached, the TTL evictor causes the expired entry to be evicted from the BackingMap. This class is used to define the TTLType value constants that determine how the the expiration time is computed for a map entry.

Since:

WAS XD 6.0, XC10

See Also:

BackingMap.setTtlEvictorType(TTLType), Serialized Form

Field Summary

static TTLType	**CREATION_TIME** A TTLType.CREATION_TIME indicates an entry expiration time is the sum of the creation time of the entry plus the "time to live" value.
static TTLType	**LAST_ACCESS_TIME** A TTLType.LAST_ACCESS_TIME indicates an entry expiration time is the sum of the last access time of the entry plus the "time to live" value.
sta	

static TTLType	**LAST_UPDATE_TIME** A TTLType.LAST_UPDATE_TIME indicates an entry expiration time is the sum of the last update time of the entry plus the "time to live" value.
static TTLType	**NONE** A TTLType.NONE indicates an entry has no expiration time and is allowed to live in the BackingMap until the application explicitly removes or invalidates the entry or a user defined evictor evicts it.

Method Summary

byte	**getId**() Get the raw value of this TTLType.
String	**toString**() Returns a string representation of the `TTLType`.
static TTLType	**valueOf**(byte id) Given the raw value of a TTLType, this method returns a TTLType object, or null if the raw value does not match an existing type.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

NONE

```
public static final TTLType NONE
```

A TTLType.NONE indicates an entry has no expiration time and is allowed to live in the BackingMap until the application explicitly removes or invalidates the entry or a user defined evictor evicts it.

CREATION_TIME

`public static final TTLType` **`CREATION_TIME`**

A TTLType.CREATION_TIME indicates an entry expiration time is the sum of the creation time of the entry plus the "time to live" value. The "time to live" value is set using the `BackingMap.setTimeToLive(int)` method and is the same for every entry and can **not** be changed by the application by using the `ObjectMap.setTimeToLive(int)` method. It can only be set prior to ObjectGrid initialization by use of the `BackingMap.setTimeToLive(int)` method.

See Also:

`BackingMap.setTimeToLive(int)`

LAST_ACCESS_TIME

`public static final TTLType` **`LAST_ACCESS_TIME`**

A TTLType.LAST_ACCESS_TIME indicates an entry expiration time is the sum of the last access time of the entry plus the "time to live" value. By default, the time to live value is set using the `BackingMap.setTimeToLive(int)` method and the default can be overridden by the application by using the `ObjectMap.setTimeToLive(int)` method.

See Also:

`BackingMap.setTimeToLive(int)`, `ObjectMap.setTimeToLive(int)`

LAST_UPDATE_TIME

`public static final TTLType` **`LAST_UPDATE_TIME`**

A TTLType.LAST_UPDATE_TIME indicates an entry expiration time is the sum of the last update time of the entry plus the "time to live" value. By default, the time to live value is set using the `BackingMap.setTimeToLive(int)` method and the default can be overridden by the application by using the `ObjectMap.setTimeToLive(int)` method. The difference between this TTLType and LAST_ACCESS_TIME is that fetch operations do not cause the entry expiration time to be updated.

Since:

7.1

See Also:

`BackingMap.setTimeToLive(int)`, `ObjectMap.setTimeToLive(int)`

Method Detail

valueOf

`public static final TTLType` **`valueOf`**`(byte id)`

Given the raw value of a TTLType, this method returns a TTLType object, or null if the raw value does not match an existing type. This method is used to deserialize this object.

Parameters:

`id` - the raw value of a TTLType

Returns:

the TTLType corresponding to the raw input value

Since:

8.6, XC10 2.5

---

getId

`public byte **getId**()`

Get the raw value of this TTLType. This method is used to serialize this object.

Returns:
the raw value of this TTLType.
Since:
8.6, XC10 2.5

---

toString

`public String **toString**()`

Returns a string representation of the `TTLType`.

Overrides:
`toString` in class `Object`

Returns:
a string representation of the `TTLType`.

---

com.ibm.websphere.objectgrid

Interface StateManager

public interface **StateManager**

The StateManager can be used to retrieve the availability state of an ObjectGrid. Use the `StateManagerFactory.getStateManager()` method to retrieve a StateManager instance.

Since:

WAS XD 6.1.0.3, XC10

Method Summary

AvailabilityState	**getObjectGridState**(ObjectGrid objectGrid) Get the AvailabilityState of an ObjectGrid.
void	**setObjectGridState**(AvailabilityState state, ObjectGrid objectGrid) Set the AvailabilityState for an ObjectGrid.

Method Detail

getObjectGridState

AvailabilityState **getObjectGridState**(ObjectGrid objectGrid)

Get the AvailabilityState of an ObjectGrid. A random shard within the ObjectGrid is chosen for reporting availability state.

Parameters:

`objectGrid` - the availability state of the specified remote ObjectGrid will be retrieved

Returns:

the AvailabilityState of the remote ObjectGrid

Throws:

`IllegalArgumentException.` - If parameter objectGrid, is either null or it is of type 'LOCAL'. See `ObjectGrid.getObjectGridType()`.

TargetNotAvailableException - if there are no active shards for the ObjectGrid specified.

---

setObjectGridState

```
void setObjectGridState(AvailabilityState state,
                        ObjectGrid objectGrid)
```

Set the AvailabilityState for an ObjectGrid. Each shard in the ObjectGrid will be transitioned to the state specified. This method does not return until each shard in the ObjectGrid has transitioned to the AvailabilityState specified or if it times-out.

Parameters:

`state` - the AvailabilityState to transition to.
`objectGrid` - the ObjectGrid to transaction to the specified AvailabilityState.

Throws:

`IllegalArgumentException.` -

1. If parameter ObjectGrid. is either null or is of type 'LOCAL'. See `ObjectGrid.getObjectGridType()`.
2. If parameter AvailabilityState is null.

---

com.ibm.websphere.objectgrid

Class SessionNotReentrantException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.TransactionException
          com.ibm.websphere.objectgrid.SessionNotReentrantException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class SessionNotReentrantException
extends TransactionException
```

A Session object can only be used by a single thread concurrently to perform map operations. If a thread tries to execute a map operation (for example, call a method on ObjectMap interface) while another thread is already executing a map operation for the Session, then this exception is thrown.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Field Summary

Fields inherited from class com.ibm.websphere.objectgrid.TransactionException
ivTransactionRolledBack

Constructor Summary

SessionNotReentrantException(String message, boolean rolledBack) Constructs a new SessionNotReentrantException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.TransactionException
isTransactionActive, wasTransactionRolledBack

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

SessionNotReentrantException

```
public SessionNotReentrantException(String message,
                                    boolean rolledBack)
```

Constructs a new SessionNotReentrantException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage(), TransactionException.wasTransactionRolledBack()

com.ibm.websphere.objectgrid

Interface Session

public interface **Session**

This interface represents a session container for `ObjectMaps`. A thread must get its own Session object to interact with ObjectGrid. You can think of this interface as a session that can only be used by a single thread at a time. A Session itself is shareable across threads so long as only one thread uses it at a time. However, if a J2EE connection/transaction infrastructure is being used, that won't be shareable across threads and will prevent the Session object from being shared across threads. A good analogy for this object is a JDBC connection to a database. For best performance, use the `close()` method to close the session once it is no longer required.

Since:

WAS XD 6.0, XC10

See Also:

`ObjectGrid.getSession()`, `ObjectGrid.getSession(Subject)`, `ObjectGrid.getSession(CredentialGenerator)`

Nested Class Summary

static class	**Session.TxCommitProtocol** The commit protocols that can be used to commit the Session's transaction

Field Summary

static long	**DEFAULT_RETRY_TIMEOUT**
static int	**TRANSACTION_NO_TIMEOUT** A special value for the timeout parameter of the `setTransactionTimeout(int)` method.

static int	**TRANSACTION_READ_COMMITTED** A transaction isolation level constant indicating that dirty reads are prevented; non-repeatable reads and phantom reads can occur.
static int	**TRANSACTION_READ_UNCOMMITTED** A transaction isolation level constant indicating that dirty reads, non-repeatable reads and phantom reads can occur.
static int	**TRANSACTION_REPEATABLE_READ** A transaction isolation level constant indicating that dirty reads and non-repeatable reads are prevented; phantom reads can occur.
static String	**TRANSACTION_TYPE_DEFAULT** A string indicating the default transaction type

Method Summary

void	**begin**() Begins a new transaction.
void	**beginNoWriteThrough**() Starts a new transaction that does not write changes through to a `Loader` or ObjectGrid server.
void	**close**() Closes this session, freeing all resources that are held.
void	**commit**() Commits a transaction.
com.ib	

m.websphere.objectgrid.query.ObjectQuery	**createObjectQuery**(String qlString) Creates an instance of an object query for executing a query over the ObjectMaps visible to this session.
void	**flush**() Forces the current changes in the Session to the Loader or ObjectGrid server.
com.ibm.websphere.objectgrid.em	**getEntityManager**() Retrieve the EntityManager associated with this Session.

.EntityManager	
ObjectMap	**getMap**(String cacheName) Returns the ObjectMap for the specified name.
ObjectGrid	**getObjectGrid**() Returns the ObjectGrid that owns this session.
long	**getRequestRetryTimeout**() Retrieves the current request retry timeout for this session.
com.ibm.websphere.objectgrid.Sessi	**getSessionHandle**() Retrieves a handle for this session.

onHandle	
int	**getTransactionIsolation**() Retrieves the current transaction isolation level for this session.
int	**getTransactionTimeout**() Gets current transaction timeout for this Session.
String	**getTransactionType**() Retrieves the transaction type that is set with the setTransactionType(String) method.
Session.TxCommitProtocol	**getTxCommitProtocol**() Retrieve the current commit protocol for this Session.
TxID	**getTxID**() Gets the `TxID` transaction identifier, if a transaction is active.
boolean	**isCommitting**() Returns whether the current session transaction is performing a session `commit()` operation.
boolean	**isFlushing**() Returns whether the current session transaction is performing a session `flush()` operation.
boo	**isMarkedRollbackOnly**() Returns whether or not the current active session transaction is marked as being

boolean	rollback only as a result of a prior call to the `markRollbackOnly(Throwable)` method on this Session.
boolean	**`isSessionHandleSet`**`()` Determines if a SessionHandle is currently set on this Session.
boolean	**`isTransactionActive`**`()` Determines if a transaction is currently active.
boolean	**`isWriteThroughEnabled`**`()` Returns whether the current session transaction is writing through to the back end `Loader` or ObjectGrid server(`true`), or if the changes are only applying to the `BackingMap` (`false`) or client respectively.
void	**`markRollbackOnly`**`(Throwable t)` Marks the current transaction as being rollback only.
void	**`processLogSequence`**`(LogSequence logSequence)` Processes a `LogSequence`.
void	**`rollback`**`()` Rolls back a transaction.
void	**`setRequestRetryTimeout`**`(long requestRetryTimeout)` Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions.
void	**`setSessionHandle`**`(com.ibm.websphere.objectgrid.SessionHandle target)` Apply a SessionHandle to this session.
void	**`setTransactionIsolation`**`(int level)` Attempts to change the transaction isolation level for this session.
void	**`setTransactionTimeout`**`(int timeout)` Sets the transaction timeout for the next transaction started by this Session object to a specified number of seconds.
void	**`setTransactionType`**`(String tranType)` Sets the transaction type for future transactions.
vo	**`setTxCommitProtocol`**`(Session.TxCommitProtocol protocol)`

id	Set the commit protocol to be used when committing this Session's transaction.
boolean	**transactionTimedOut**() Determines whether the current session transaction has timed out.

Field Detail

TRANSACTION_TYPE_DEFAULT

static final String **TRANSACTION_TYPE_DEFAULT**

A string indicating the default transaction type

See Also:
Constant Field Values

TRANSACTION_NO_TIMEOUT

static final int **TRANSACTION_NO_TIMEOUT**

A special value for the timeout parameter of the `setTransactionTimeout(int)` method. This special value is used to indicate that the next transaction started by this Session is allowed unlimited amount of time.

See Also:
`setTransactionTimeout(int)`, Constant Field Values

DEFAULT_RETRY_TIMEOUT

static final long **DEFAULT_RETRY_TIMEOUT**

See Also:
Constant Field Values

TRANSACTION_REPEATABLE_READ

static final int **TRANSACTION_REPEATABLE_READ**

A transaction isolation level constant indicating that dirty reads and non-repeatable reads are prevented; phantom reads can occur. This level prohibits a transaction from reading an uncommitted cache entry, and it also prohibits the situation where one transaction reads an entry, a second transaction alters the entry, and the first transaction rereads the entry, getting different values the second time (a "non-repeatable read").

Since:
WAS XD 6.1.0.1
See Also:
`setTransactionIsolation(int)`, Constant Field Values

TRANSACTION_READ_COMMITTED

```
static final int TRANSACTION_READ_COMMITTED
```

A transaction isolation level constant indicating that dirty reads are prevented; non-repeatable reads and phantom reads can occur. This level only prohibits a transaction from reading a cache entry with uncommitted changes in it.

Since:
WAS XD 6.1.0.1
See Also:
`setTransactionIsolation(int)`, Constant Field Values

TRANSACTION_READ_UNCOMMITTED

```
static final int TRANSACTION_READ_UNCOMMITTED
```

A transaction isolation level constant indicating that dirty reads, non-repeatable reads and phantom reads can occur. This level allows a cache entry changed by one transaction to be read by another transaction before any changes in that entry have been committed (a "dirty read"). If any of the changes are rolled back, the second transaction will have retrieved an invalid entry.

Since:
WAS XD 6.1.0.1
See Also:
`setTransactionIsolation(int)`, Constant Field Values

Method Detail

beginNoWriteThrough

```
void beginNoWriteThrough()
                      throws TransactionAlreadyActiveException,
                             TransactionException
```

Starts a new transaction that does not write changes through to a `Loader` or ObjectGrid server.

Changes made by the session transaction started by this method are only applied to the `BackingMap` and not given to the `Loader`. This method can be used to apply changes made in a peer cache to the local `BackingMap` only. In addition, with a distributed map, this method can be used to start a session transaction which changes will only be applied to the client `BackingMap`, but not the `BackingMap` on the server side.

Throws:
`TransactionAlreadyActiveException` - if there is already an active transaction
`TransactionException` - a `TransactionCallbackException` occurred or some other error occurred starting a new transaction

---

getMap

```
ObjectMap getMap(String cacheName)
                 throws UndefinedMapException
```

Returns the ObjectMap for the specified name.

The ObjectMap is used to retrieve and modify values in the BackingMap. Multiple invocations of this method on the same Session object will always return the same object.

This method can also be used to create a BackingMap and its associated ObjectGrid after ObjectGrid initialization. If cacheName does not match the name of a previously created

map, a name comparison will be executed against template maps that have been configured. The ObjectMap and BackingMap will be created if the name matches the regular expression of a template.

Required Client Permission: `ObjectGridPermission.DYNAMIC_MAP` (when creating a new map from a template)

Parameters:

`cacheName` - name of desired map

Returns:

ObjectMap the transactional interface to modify values in the map

Throws:

`UndefinedMapException` - if the map is not defined.

See Also:

`ObjectGrid.defineMap(String)`, `ObjectMap`

---

begin

```
void begin()
          throws TransactionAlreadyActiveException,
                 TransactionException
```

Begins a new transaction.

Throws:

`TransactionAlreadyActiveException` - if this method is invoked with an active transaction

`TransactionException` - a `TransactionCallbackException` occurred or some other error occurred starting a new transaction

---

commit

```
void commit()
          throws NoActiveTransactionException,
                 TransactionException
```

Commits a transaction.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

`TransactionException` - if an error occurred during commit processing, see the caused by to determine the root error

See Also:

`markRollbackOnly(Throwable)`

---

rollback

```
void rollback()
          throws NoActiveTransactionException,
                 TransactionException
```

Rolls back a transaction.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

`TransactionException` - if an error occurred during rollback processing, see the caused by to determine the root error

---

flush

```
void flush()
        throws NoActiveTransactionException,
               TransactionException
```

Forces the current changes in the Session to the `Loader` or ObjectGrid server. This method does not commit the changes, it just applies the changes.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction
`TransactionException` - if an error occurred during flush processing, see the caused by to determine the root error

---

getObjectGrid

```
ObjectGrid getObjectGrid()
```

Returns the ObjectGrid that owns this session.

Returns:

the owning ObjectGrid instance.

---

isTransactionActive

```
boolean isTransactionActive()
```

Determines if a transaction is currently active.

Returns:

true if a transaction is currently active for this session.

Since:

WAS XD 6.1 FIX3

---

getTxID

```
TxID getTxID()
        throws NoActiveTransactionException
```

Gets the `TxID` transaction identifier, if a transaction is active.

Returns:

The current TxID object.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

---

isWriteThroughEnabled

```
boolean isWriteThroughEnabled()
```

Returns whether the current session transaction is writing through to the back end `Loader` or ObjectGrid server(`true`), or if the changes are only applying to the `BackingMap` (`false`) or client respectively.

Returns:

`true`, if write through is enabled

See Also:

`begin()`, `beginNoWriteThrough()`

---

setTransactionType

```
void setTransactionType(String tranType)
```

Sets the transaction type for future transactions.

After this method is called, all future transactions will have the same type until another transaction type is set. If no transaction type is set, the default transaction type `TRANSACTION_TYPE_DEFAULT` will be used.

Transaction types are used mainly for statistical data tracking purpose. Users can predefine types of transactions that will be executed in an application. The idea is to categorize transactions with the same characteristics to one category (type), so one transaction response time statistics can be used to track each transaction type. This approach is useful when your application has different types of transactions. Some types of transactions, such as update transactions, process longer than others transactions, such as read-only transactions. By using the transaction type, different transactions are tracked by different statistics, so the statistics can be more useful.

Parameters:

`tranType` - the transaction type for future transactions.

See Also:

`TRANSACTION_TYPE_DEFAULT`

---

getTransactionType

```
String getTransactionType()
```

Retrieves the transaction type that is set with the setTransactionType(String) method.

Returns:

the transaction type for the session.

Since:

7.1.1.1

---

processLogSequence

```
void processLogSequence(LogSequence logSequence)
                        throws NoActiveTransactionException,
                               UndefinedMapException,
                               ObjectGridException
```

Processes a `LogSequence`.

Each `LogElement` within the `LogSequence` will be examined and the appropriate operation (insert, update, invalidate, etc) will be performed against the `BackingMap` identified by the `LogSequence`'s map name. An ObjectGrid Session must be active before this method is invoked. The caller is responsible for issuing the appropriate commit or rollback invocation to complete the Session. Autocommit processing is not available for this method invocation.

The main use of this method is for processing a `LogSequence` that was received by a remote JVM. For example, using the Distributed Commit support, the `LogSequences` associated with a given committed Session are distributed to other listening `ObjectGrids` in other JVMs. After receiving the `LogSequences` at the remote JVM, the listener could start a Session using `beginNoWriteThrough()`, invoke this method, and commit the Session transaction.

Parameters:

`logSequence` - LogSequence of changes to be applied to an active transaction

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

`UndefinedMapException` - if the map referenced by the LogSequence cannot be found

ObjectGridException - if the LogSequence elements cannot be processed

See Also:

beginNoWriteThrough(), LogSequence, ObjectGridEventListener

---

isFlushing

`boolean` **`isFlushing`**`()`

Returns whether the current session transaction is performing a session `flush()` operation. It is helpful to know if a session `flush()` is active (`true`), or if only an `ObjectMap.flush()` is in progress (returns `false` in this case).

Returns:

true, if the session is executing a session `flush()` call.

Since:

WAS XD 6.0.1

See Also:

flush(), ObjectMap.flush()

---

isCommitting

`boolean` **`isCommitting`**`()`

Returns whether the current session transaction is performing a session `commit()` operation. It is helpful to know if a session commit is active (`true`), or if an `ObjectMap.flush()` or session `flush()` is in progress (returns `false` in these cases).

Returns:

true, if session is executing a session `commit()` call.

Since:

WAS XD 6.0.1

See Also:

commit(), flush(), ObjectMap.flush()

---

markRollbackOnly

`void` **`markRollbackOnly`**`(Throwable t)`
`throws NoActiveTransactionException`

Marks the current transaction as being rollback only.

Marking a transaction rollback only ensures that even if the `commit()` method is called for this session transaction, the transaction is rolled back. A rollback is typically done when either ObjectGrid itself or the application knows that data corruption could occur if the `commit()` method was allowed to commit the transaction. Once this method is called, the Throwable object that is passed to it is chained to the `TransactionException` that is thrown if the commit method is ever called. Any subsequent calls to this method for the current active transaction is ignored (e.g. only the first call that passes a non null Throwable reference is used). Once the transaction is completed, the rollback only mark is removed so that the next transaction that is started using this session can be committed.

Parameters:

`t` - the Throwable that caused this method to be called.

Throws:

NoActiveTransactionException - if there is no active transaction for this Session.

Since:

WAS XD 6.0.1

See Also:

commit(), TransactionException

isMarkedRollbackOnly

`boolean` **`isMarkedRollbackOnly`**`()`

Returns whether or not the current active session transaction is marked as being rollback only as a result of a prior call to the `markRollbackOnly(Throwable)` method on this Session.

Returns:
true if and only if current session transaction is marked rollback only.
Since:
WAS XD 6.0.1
See Also:
`markRollbackOnly(Throwable)`

---

setTransactionTimeout

`void` **`setTransactionTimeout`**`(int timeout)`

Sets the transaction timeout for the next transaction started by this Session object to a specified number of seconds.

This method does not affect the transaction timeout of any transactions previously started by this Session. It only affects transactions that are started after this method is called. If this method is never called, the ObjectGrid configured transaction timeout value is used.

Parameters:
`timeout` - is the transaction timeout value in seconds. Use the special value `TRANSACTION_NO_TIMEOUT` if transaction is allowed unlimited amount of time and no transaction timeout should occur.
Since:
WAS XD 6.0.1
See Also:
`TRANSACTION_NO_TIMEOUT`, `ObjectGrid.setTxTimeout(int)`, `TransactionTimeoutException`

---

getTransactionTimeout

`int` **`getTransactionTimeout`**`()`

Gets current transaction timeout for this Session.

The transaction timeout value returned is the value that was configured for the `ObjectGrid` using `ObjectGrid.setTxTimeout(int)` or the value passed to `setTransactionTimeout(int)` to override the value configured on `ObjectGrid`. The return value is in seconds.

Returns:
timeout value in seconds.
Since:
WAS XD 6.0.1
See Also:
`setTransactionTimeout(int)`, `ObjectGrid.setTxTimeout(int)`

---

getTransactionIsolation

`int` **`getTransactionIsolation`**`()`

Retrieves the current transaction isolation level for this session.

Returns:
one of the following `Session` constants: `TRANSACTION_READ_UNCOMMITTED`, `TRANSACTION_READ_COMMITTED` or `TRANSACTION_REPEATABLE_READ`

Since:
WAS XD 6.1.0.1

---

transactionTimedOut

`boolean` **`transactionTimedOut`**`()`

Determines whether the current session transaction has timed out.

Returns:
true if and only if transaction has timed out.

Since:
WAS XD 6.0.1

See Also:
`setTransactionTimeout(int)`

---

createObjectQuery

`com.ibm.websphere.objectgrid.query.ObjectQuery` **`createObjectQuery`**`(String qlString) throws com.ibm.websphere.objectgrid.query.ObjectQueryException`

Creates an instance of an object query for executing a query over the ObjectMaps visible to this session.

When ObjectGrid security is enabled, this method requires an com.ibm.websphere.objectgrid.security.ObjectGridPermission with action "query".

Required Client Permission: `ObjectGridPermission.QUERY`

Parameters:
`qlString` - a query string

Returns:
the new query instance.

Throws:
`com.ibm.websphere.objectgrid.query.ObjectQueryException` - if an error occurs creating the object query.

Since:
WAS XD 6.1

---

getEntityManager

`com.ibm.websphere.objectgrid.em.EntityManager` **`getEntityManager`**`()`

Retrieve the EntityManager associated with this Session. Each session is associated with a single EntityManager instance. Repeated calls to this method on the same Session instance will result in the same EntityManager instance.

Returns:
this session's EntityManager instance.

Since:
WAS XD 6.1

---

setTransactionIsolation

```
void setTransactionIsolation(int level)
```

Attempts to change the transaction isolation level for this session. The constants defined in the `Session` interface are the possible transaction isolation levels.

This method should normally be invoked prior to beginning a transaction. Invoking after a transaction has started may result in an exception.

Parameters:

`level` - one of the following `Session` constants: `TRANSACTION_READ_UNCOMMITTED`, `TRANSACTION_READ_COMMITTED` or `TRANSACTION_REPEATABLE_READ`

Since:

WAS XD 6.1.0.1

---

getSessionHandle

```
com.ibm.websphere.objectgrid.SessionHandle getSessionHandle()
```

Retrieves a handle for this session.

A SessionHandle contains partition information for the current session and can be re-applied to a new session using the `setSessionHandle(SessionHandle)` method. A SessionHandle is only applicable for ObjectGrids using per-container partition placement. If `setSessionHandle(SessionHandle)` is not called before invoking this method, a Session Handle is selected using the properties configured in the `ClientProperties`. If there are no per-container partition placement mapsets or more than one in the ObjectGrid, an `IllegalStateException` is thrown.

Returns:

the SessionHandle for this session

Throws:

`IllegalStateException` - if this method is called in an invalid environment.

Since:

WAS XD 6.1.0.3

---

setSessionHandle

```
void setSessionHandle(com.ibm.websphere.objectgrid.SessionHandle target)
                      throws TargetNotAvailableException
```

Apply a SessionHandle to this session.

Parameters:

`target` - the SessionHandle to apply or null to disassociate a SessionHandle from this session.

Throws:

`TargetNotAvailableException` - when the target is no longer available.

`IllegalStateException` - if the Session has modified some maps already and the SessionHandle has already been set or if this method is called in an invalidate environment.

Since:

WAS XD 6.1.0.3

---

setRequestRetryTimeout

```
void setRequestRetryTimeout(long requestRetryTimeout)
```

Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions. A request will timeout

when either the request timeout expires or the transaction timeout expires, whichever expires first.

A value of 0 indicates that all requests should fail immediately and avoid any retry logic. Exceptions that cannot succeed even if tried again such as DuplicateKeyException exceptions will be thrown immediately.

A value of -1 indicates that the request retry timeout is not set, meaning that the request duration is governed by the request retry timeout set on the ClientProperties. If the ClientProperties is also set to -1, then the request retry timeout is governed by the transaction timeout.

Parameters:

`requestRetryTimeout` - the duration in milliseconds retry a client request, 0 if the request should fail immediately or -1 if the request timeout is not set.

Since:

7.0

See Also:

`ClientProperties.setRequestRetryTimeout(long)`, `setTransactionTimeout(int)`

---

getRequestRetryTimeout

`long` **`getRequestRetryTimeout`**`()`

Retrieves the current request retry timeout for this session. Returns -1 if it was not set.

Returns:

the duration in milliseconds retry a client request, 0 if the request should fail immediately or -1 if the request timeout is not set.

Since:

7.0

---

isSessionHandleSet

`boolean` **`isSessionHandleSet`**`()`

Determines if a SessionHandle is currently set on this Session.

Returns:

true if a SessionHandle is currently set on this session.

Since:

7.1

---

close

`void` **`close`**`()`

Closes this session, freeing all resources that are held. Once closed, this session must be discarded. Use one of the `ObjectGrid.getSession()` methods to retrieve a new session. If the session has an active transaction, the transaction will be rolled back and the session resources are not freed.

Throws:

`ObjectGridRuntimeException` - thrown if there is a problem releasing resources held by this session.

Since:

7.1.1

---

setTxCommitProtocol

`void` **`setTxCommitProtocol`**`(`[`Session.TxCommitProtocol`](#) `protocol)`

Set the commit protocol to be used when committing this Session's transaction. The constants defined in the `TxCommitProtocol` enum are the possible commit protocols.

This method should normally be invoked prior to beginning a transaction. Invoking after a transaction has started will result in an exception.

Parameters:

`protocol` - one of the following constants `TxCommitProtocol.ONEPHASE` or `TxCommitProtocol.TWOPHASE`

Since:

8.6, XC10 2.5

---

getTxCommitProtocol

[`Session.TxCommitProtocol`](#) **`getTxCommitProtocol`**`()`

Retrieve the current commit protocol for this Session.

Returns:

one of the following constants `TxCommitProtocol.ONEPHASE` or `TxCommitProtocol.TWOPHASE`

Since:

8.6, XC10 2.5

---

com.ibm.websphere.objectgrid

Enum Session.TxCommitProtocol

java.lang.Object
  java.lang.Enum<Session.TxCommitProtocol>
    com.ibm.websphere.objectgrid.Session.TxCommitProtocol

All Implemented Interfaces:

Serializable, Comparable<Session.TxCommitProtocol>

Enclosing interface:

Session

```
public static enum Session.TxCommitProtocol
extends Enum<Session.TxCommitProtocol>
```

The commit protocols that can be used to commit the Session's transaction

Since:

8.6, XC10 2.5

Enum Constant Summary

ONEPHASE A commit protocol constant indicating that the Session transaction can read from multiple partitions but can only write to a single partition.
TWOPHASE A commit protocol constant indicating that the Session transaction can read and write from multiple partitions.

Method Summary

static Session.TxCommitProtocol	**valueOf**(String name) Returns the enum constant of this type with the specified name.

Protocol	
staticSession.TxCommitProtocol[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

ONEPHASE

public static final Session.TxCommitProtocol **ONEPHASE**

A commit protocol constant indicating that the Session transaction can read from multiple partitions but can only write to a single partition. A TransactionException is thrown if the transaction writes to multiple partitions. The transaction is committed using the one-phase commit protocol.

TWOPHASE

public static final Session.TxCommitProtocol **TWOPHASE**

A commit protocol constant indicating that the Session transaction can read and write from multiple partitions. The transaction is committed using the two-phase commit protocol. If the transaction only writes to a single partition then the transaction is

committed using the one-phase commit protocol.

Method Detail

values

```
public static Session.TxCommitProtocol[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (Session.TxCommitProtocol c : Session.TxCommitProtocol.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static Session.TxCommitProtocol valueOf(String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
`name` - the name of the enum constant to be returned.
Returns:
the enum constant with the specified name
Throws:
`IllegalArgumentException` - if this enum type has no constant with the specified name
`NullPointerException` - if the argument is null

---

com.ibm.websphere.objectgrid

Class ReplicationVotedToRollbackTransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.TransactionCallbackException
          com.ibm.websphere.objectgrid.ReplicationVotedToRollbackTransactionException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **ReplicationVotedToRollbackTransactionException**
extends TransactionCallbackException

This exception is thrown when a transaction was rolled back because some/all of the replicas failed to apply the transaction when in synchronous replication mode.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

ReplicationVotedToRollbackTransactionException() Constructs a new ReplicationVotedToRollbackTransactionException with `null` as its detail message.
ReplicationVotedToRollbackTransactionException(String message) Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message.
ReplicationVotedToRollbackTransactionException(String message, Throwable cause) Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message and cause.
ReplicationVotedToRollbackTransactionException(Throwable cause) Constructs a new ReplicationVotedToRollbackTransactionException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException()
```

Constructs a new ReplicationVotedToRollbackTransactionException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException(String message)
```

Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
ObjectGridException.initCause(Throwable), Throwable.getMessage()

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException(String message,
                                                      Throwable cause)
```

Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ReplicationVotedToRollbackTransactionException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
ObjectGridException.getCause(), Throwable.getMessage()

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException(Throwable cause)
```

Constructs a new ReplicationVotedToRollbackTransactionException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ReplicationVotedToRollbackTransactionExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Class ReadOnlyException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.ReadOnlyException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class ReadOnlyException
extends ObjectGridException
```

This exception is thrown when an attempt is made to modifying operations on a read only maps.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

ReadOnlyException() Constructs a new ReadOnlyException with `null` as its detail message.
ReadOnlyException(String message) Constructs a new ReadOnlyException with the specified detail message.
ReadOnlyException(String message, Throwable cause) Constructs a new ReadOnlyException with the specified detail message and cause.
ReadOnlyException(Throwable cause) Constructs a new ReadOnlyException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ReadOnlyException

```
public ReadOnlyException()
```

Constructs a new ReadOnlyException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

ReadOnlyException

```
public ReadOnlyException(String message)
```

Constructs a new ReadOnlyException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

ReadOnlyException

```
public ReadOnlyException(String message,
                         Throwable cause)
```

Constructs a new ReadOnlyException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ReadOnlyException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

ReadOnlyException

```
public ReadOnlyException(Throwable cause)
```

Constructs a new ReadOnlyException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ReadOnlyExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Interface PartitionManager

public interface **PartitionManager**

This interface will be used for calculating the proper partition for a given input key. The set of partitions is determined by the BackingMap configuration.

Since:
WAS XD 6.0.1, XC10

See Also:
BackingMap.getPartitionManager()

Method Summary

int	**getNumOfPartitions**() Returns the number of configured partitions for this PartitionManager.
int	**getPartition**(Object key) Obtains a 0-based partition number determined by the input parameter's hashCode() method.
List	**getPartitionLists**(List keyList) This method is very similar to `getPartitions(List)`, except it returns the keys organized by the partition identifiers.
List	**getPartitions**(List keyList) Obtains the 0-based partition numbers for each of the keys in the input `List` of keys.
List	**partitionLogSequence**(LogSequence ls) Partitions a `LogSequence` based on the partitioning algorithm for the Map.

Method Detail

getPartition

int **getPartition**(Object key)

Obtains a 0-based partition number determined by the input parameter's hashCode() method.

Parameters:
`key` - Individual key used to determine partition (can not be null)

Returns:
int 0-based partition number

getPartitions

List **getPartitions**(List keyList)

Obtains the 0-based partition numbers for each of the keys in the input `List` of keys. Each object in the returned list of partition identifiers is an instance of `java.lang.Integer`.

Parameters:
`keyList` - Ordered list of keys
Returns:
List of partition identifiers that corresponds to the input list of keys

---

getPartitionLists

List **getPartitionLists**(List keyList)

This method is very similar to `getPartitions(List)`, except it returns the keys organized by the partition identifiers. The return value is a `List` of `Lists`. The outer `List` is an ordered `List` of the partition numbers, with the first entry in the `List` corresponding to partition 0. The inner `Lists` contain the keys from the input parameter that correspond to that partition identifier.

The return value will always contain a `List` object. Either the outer or the inner `Lists` may contain zero elements, but the `List` objects themselves will not be `null`.

Parameters:
`keyList` - Ordered list of keys
Returns:
List of Lists that correspond to the 0-based partition numbers, with each inner List containing the set of keys that parse to that partition number.

---

partitionLogSequence

List **partitionLogSequence**(LogSequence ls)

Partitions a `LogSequence` based on the partitioning algorithm for the Map.

Parameters:
`ls` - LogSequence that needs to be partitioned
Returns:
List of partitioned LogSequences. The first LogSequence in the List corresponds to the first partition, etc.

---

getNumOfPartitions

int **getNumOfPartitions**()

Returns the number of configured partitions for this PartitionManager.

Returns:
the number of configured partitions

---

com.ibm.websphere.objectgrid

Interface ObjectMap

public interface **ObjectMap**

This is a handle to a named Map. Maps should have homogeneous keys and values. An instance of this ObjectMap can only be used by the thread that is currently associated with the Session that was used to get this ObjectMap instance. Both Session and ObjectMap objects are not allowed to be shared by multiple threads concurrently.

The `CopyMode` setting on the map determines whether or not a copy of the value is returned by get methods. It also determines whether or not a copy of the committed value is made at commit time. The `LockStrategy` setting for the map determines whether or not a lock is obtained for each map entry accessed by the transaction, the lock mode of the lock obtained, and when the lock is obtained.

Each data access method includes a "Specification details" table that includes the following information:

Required permission:	The permission required to use the API.
Pessimistic read lock acquired:	The type of lock that is acquired when using pessimistic locking with repeatable read transaction isolation.
Pessimistic read lock held:	The type of lock that is held for the duration of the transaction when using pessimistic locking with repeatable read transaction isolation. Locks can be upgraded but not demoted during a transaction.
Transaction:	The state of the transaction when invoking the API. • **Manual** - The transaction is explicitly demarcated using the associated `Session` and the specified API is used within the scope of that transaction. • **Automatic** - The transaction is automatically demarcated. This is also referred to as an auto-commit transaction or API which the associated `Session.isTransactionActive()` is false when invoking the API.
Cache tier:	Identifies the map cache tiers that are included when fetching or updating cache entries in the call and under what circumstances. The following tiers are available for client maps: • Transactional Cache • Near Cache (if enabled) • Server Cache • Loader (if enabled) The following tiers are available for local maps: • Transactional Cache • Local Cache • Loader (if enabled)

Since:

WAS XD 6.0, XC10

See Also:

Session.getMap(String), BackingMap.setCopyMode(CopyMode, Class), BackingMap.setLockStrategy(LockStrategy)

Nested Class Summary

static class	**ObjectMap.PutMode** Identifies the operation mode of the put(Object, Object), putAll(Map) , JavaMap.put(Object, Object) and JavaMap.putAll(Map) methods.

Field Summary

static long	**QUEUE_TIMEOUT_INFINITE** Used as a parameter on the getNextKey(long) method, specifies the method should block until a key becomes available.
static long	**QUEUE_TIMEOUT_NONE** Used as a parameter on the getNextKey(long) method, specifies to return a null value if the map is empty.
static int	**TTL_FOREVER** A constant indicating the time-to-live value is "forever".
static int	**USE_DEFAULT** A constant indicating the time-to-live value or lock timeout value is the default setting.

Method Summary

v	**clear**()

void	Clear all keys from the Map.
void	**clearCopyMode**() Resets the CopyMode back to the one in the BackingMap.
boolean	**containsKey**(Object key) Returns `true` if this map contains a mapping for the specified key.
void	**flush**() Pushes the current set of changes for the `ObjectMap` instance to the `Loader` without committing the changes.
Object	**get**(Object key) Retrieves the object from the cache at the given key.
com.ibm.websphere.objectgrid.datagrid.AgentMan	**getAgentManager**() Returns the Agent Manager that allows DataGrid operations to be performed on the objects within this Map.

a g e r	
List	**getAll**(List keyList) Gets a list of entries from the map.
List	**getAllForUpdate**(List keyList) Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries.
c o m . i b m . w e b s p h e r e . p r o j e c t o r . m d . E n t i t y M e t a d a t a	**getEntityMetadata**() Retrieve the metadata for the entity associated with this map.
Object	**getForUpdate**(Object key) Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry.
Object	**getIndex**(String name) Returns a reference to the named index that can be used with this Map.

Object	**getIndex**(String name, boolean forUpdate) Returns a reference to the named index that can be used with this Map.
Map	**getJavaMap**() Returns an implementation of `java.util.Map` that is backed by this `ObjectMap`.
com.ibm.websphere.objectgrid.OutputFormat	**getKeyOutputFormat**() Retrieves the data format for all data access APIs that return cache keys for the current session.
int	**getMapType**() Returns the type of the underlying BackingMap.
String	**getName**() Returns the name of the ObjectMap as defined by the configuration.
Object	**getNextKey**(long timeout) Retrieves a key off the map in first-in-first-out (FIFO) insert order.
Ob	

int	**getPutMode**() Retrieve the current PutMode set for this ObjectMap instance.
com.ibm.websphere.objectgrid.OutputFormat	**getValueOutputFormat**() Retrieves the data format for all data access APIs that return cache values for the current session.
void	**insert**(Object key, Object value) Performs an explicit insert of a given entry.
void	**invalidate**(Object key, boolean isGlobal) Invalidates an entry in the cache based on the key parameter.
void	**invalidateAll**(Collection keyList, boolean isGlobal) Invalidate a set of cache entries based on the Collection of keys provided.
bo	

boolean	**lock**(Object key, com.ibm.websphere.objectgrid.LockMode lockMode) Obtains a lock for the given key.
List<Boolean>	**lockAll**(List keyList, com.ibm.websphere.objectgrid.LockMode lockMode) Obtains locks for the given keys.
Object	**put**(Object key, Object value) Puts the Object value into the cache at location represented by key.
void	**putAll**(Map map) Puts each of the Object value into the cache at location represented by key contained in the Map.
Object	**remove**(Object key) Removes the Object value from the cache represented by key.
void	**removeAll**(Collection keyList) Batch remove from the Map.
void	**setCopyMode**(CopyMode copyMode, Class valueInterface) Allows the CopyMode for the Map to be overridden on this map on this session only.
void	**setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data format for all data access APIs that return cache keys for the current session.
void	**setLockTimeout**(int seconds) Overrides the BackingMap's lock timeout for this ObjectMap.
void	**setPutMode**(ObjectMap.PutMode putMode) Allows the default operation for the put(Object, Object) and putAll(Map) methods to be changed, allowing the put to use the optimized upsert(Object, Object) and upsertAll(LinkedHashMap) implementations.
int	**setTimeToLive**(int ttl) Establishes the number of seconds that any given cache entry can live for, which is referred to as "time to live" or TTL.
void	**setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data format for all data access APIs that return cache values for the current session.

void	**touch**(Object key) Updates the last access time in the BackingMap without retrieving the value to the ObjectMap.
void	**update**(Object key, Object value) Performs an explicit update of a given entry.
void	**upsert**(Object key, Object value) Puts the Object value into the cache at location represented by key.
void	**upsertAll**(LinkedHashMap map) Puts each of the Object value into the cache at location represented by key contained in the Map.

Field Detail

TTL_FOREVER

```
static final int TTL_FOREVER
```

A constant indicating the time-to-live value is "forever".

See Also:
Constant Field Values

USE_DEFAULT

```
static final int USE_DEFAULT
```

A constant indicating the time-to-live value or lock timeout value is the default setting.

The default setting is to retain the time-to-live value for any existing map entry and to use the default value from BackingMap setting if a new map entry is being created.

For lock timeout override the default setting is to use the value defined on the BackingMap

See Also:
setLockTimeout(int), setTimeToLive(int), BackingMap.setTimeToLive(int), BackingMap.getTimeToLive(), BackingMap.setLockTimeout(int), Constant Field Values

---

QUEUE_TIMEOUT_NONE

```
static final long QUEUE_TIMEOUT_NONE
```

Used as a parameter on the getNextKey(long) method, specifies to return a null value if the map is empty.

See Also:
Constant Field Values

QUEUE_TIMEOUT_INFINITE

`static final long` **`QUEUE_TIMEOUT_INFINITE`**

Used as a parameter on the `getNextKey(long)` method, specifies the method should block until a key becomes available.

See Also:
Constant Field Values

Method Detail

getName

`String` **`getName`**`()`

Returns the name of the ObjectMap as defined by the configuration.

Returns:
name of ObjectMap

---

get

`Object` **`get`**`(Object key)`
`throws ObjectGridException`

Retrieves the object from the cache at the given key.

Whether or not a copy of the object is returned is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if a value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`
Pessimistic read lock held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found.

Parameters:
`key` - The entry to fetch
Returns:
the value, `null`, `SerializedValue` or `Tuple`
Throws:
`IllegalArgumentException` - if key is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.
See Also:
`containsKey(Object)`, `getForUpdate(Object)`, `CopyMode`

---

put

```
Object put(Object key,
           Object value)
           throws ObjectGridException
```

Puts the Object value into the cache at location represented by key.

The value will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the `setPutMode(PutMode)` property:

`ObjectMap.PutMode.INSERTUPDATE`	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
`ObjectMap.PutMode.UPSERT`	The value is put into the map using the specification of the `upsert(Object, Object)`.

Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.WRITE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - The entry to put into the map
`value` - The value to put into the map using the key

Returns:

If `ObjectMap.PutMode.INSERTUPDATE` is set, return the previous value in this transaction.
If `ObjectMap.PutMode.UPSERT` is set, the return value is null.

Throws:

`IllegalArgumentException` - if key is `null`, or if the map does not allow `null` values and value is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`CopyMode`

---

getForUpdate

```
Object getForUpdate(Object key)
                    throws ObjectGridException
```

Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry. See `LockStrategy.PESSIMISTIC` for additional information. Whether or not a copy of the object is returned is determined by the `CopyMode` setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if the value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.UPGRADABLE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

Parameters:

`key` - The entry to fetch

Returns:

the value, `null`, `SerializedValue` or `Tuple`

Throws:

`IllegalArgumentException` - if key is `null`

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`containsKey(Object)`, `get(Object)`, `CopyMode`, `LockStrategy.PESSIMISTIC`

---

remove

```
Object remove(Object key)
              throws ObjectGridException
```

Removes the Object value from the cache represented by key.

This removal will be pushed down to the BackingMap/Loader at commit time. If the key cannot be found in the map, a null value will be returned.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permissi	`MapPermission.REMOVE`

on:	
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - The entry to remove

Returns:

the current value at invocation time

Throws:

`IllegalArgumentException` - if key is `null`

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

getAll

```
List getAll(List keyList)
            throws ObjectGridException
```

Gets a list of entries from the map.

If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`
Pessimistic read lock held:	Yes
Transaction:	Automatic or manual
Cache tier:	For each key, progresses to all tiers until the key is found.

Parameters:

`keyList` - A list of keys for identifying which entries to fetch

Returns:

a list of values

Throws:

`IllegalArgumentException` - if keyList is null or contains a `null` element.

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`get(Object)`

---

getAllForUpdate

```
List getAllForUpdate(List keyList)
                     throws ObjectGridException
```

Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries. See `LockStrategy.PESSIMISTIC` for additional information. If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.UPGRADABLE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the keys are found and the appropriate locks are acquired.

Parameters:

`keyList` - A list of keys for identifying which entries to fetch

Returns:

a list of values

Throws:

`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`getAll(List)`, `getForUpdate(Object)`, `LockStrategy.PESSIMISTIC`

---

removeAll

```
void removeAll(Collection keyList)
               throws ObjectGridException
```

Batch remove from the Map. If a key in the list cannot be found, it will be ignored.

Specification details:

Required permission:	`MapPermission.REMOVE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`keyList` - A list of keys for identifying which entries to remove

Throws:

IllegalArgumentException - if keyList is null or contains a null element.
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

remove(Object)

---

putAll

```
void putAll(Map map)
            throws ObjectGridException
```

Puts each of the Object value into the cache at location represented by key contained in the Map.

The values will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the setPutMode(PutMode) property:

ObjectMap.PutMode.INSERTUPDATE	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
ObjectMap.PutMode.UPSERT	The values are put into the map using the specification of the upsertAll(LinkedHashMap).

Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See CopyMode for a description of each possible copy mode.

An existing Map object will be passed in to use for obtaining the keys and values to be inserted or updated into the existing Map.

Specification details:

Required permission:	MapPermission.WRITE
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use Session.beginNoWriteThrough() to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

map - The key/values to be put into the map.

Throws:

IllegalArgumentException - if map is null or contains a null key or if null values are not allowed and map contains a null value.
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

put(Object, Object)

---

invalidate

```
void invalidate(Object key,
                boolean isGlobal)
                throws ObjectGridException
```

Invalidates an entry in the cache based on the key parameter.

If the key's value has changes pending in the ObjectMap, it is the application's responsibility to flush these changes to the Loader before invalidation. If a flush is not performed prior to invoking the invalidate operation, all pending changes for this key will be removed from the ObjectMap. If the key cannot be found in the map, it will be ignored.

The isGlobal parameter is used to indicate which cache level is used to invalidate the entries. If isGlobal is true, when the transaction is committed, the key is removed from the BackingMap also. If a subsequent get operation is performed, the BackingMap will be skipped and the Loader will be used to get the data. If isGlobal is false, the entry is only invalidated in the ObjectMap (transactional cache). If a subsequent get operation is performed, the BackingMap can be used; and, if it's not in the BackingMap, the Loader will be used to get the data.

A typical use of isGlobal being false is when a large number of records are touched in a transaction and the application wants to evict records that are no longer used in the cache.

Specification details:

Required permission:	`MapPermission.INVALIDATE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit except the Loader. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps. Set the `isGlobal` parameter to `false` to limit the operation to the transaction cache tier.

Parameters:

`key` - Object representing the key to be used for cache entry invalidation
`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the ObjectMap (false).

Throws:

`IllegalArgumentException` - if key is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

invalidateAll

```
void invalidateAll(Collection keyList,
                   boolean isGlobal)
                   throws ObjectGridException
```

Invalidate a set of cache entries based on the Collection of keys provided. If a key in the collection cannot be found, it will be ignored.

Specification details:

Required permission:	`MapPermission.INVALIDATE`
Transaction:	Automatic or manual

actio
n:

Cach
e tier: Applied to all tiers during commit except the Loader. Use [Session.beginNoWriteThrough()](#) to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps. Set the `isGlobal` parameter to `false` to limit the operation to the transaction cache tier.

Parameters:

`keyList` - A Collection of keys representing the entries to be invalidated

`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the ObjectMap (false).

Throws:

[IllegalArgumentException](#) - if keyList is null or contains a `null` element.

[ObjectGridException](#) - if an error occurs during processing

[AccessControlException](#) - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

[invalidate(Object, boolean)](#)

---

setTimeToLive

`int` **`setTimeToLive`**`(int ttl)`

Establishes the number of seconds that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this `ObjectMap`, any previous value set by the `BackingMap.setTimeToLive(int)` method is overridden for this ObjectMap. If this method is never called on the ObjectMap, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from BackingMap setting if a new map entry is being created. If TTL is never set on the BackingMap, the cache entry can live "forever".

This method can only be used when the `TTLType` is set to `LAST_ACCESS_TIME` or `LAST_UPDATE_TIME` on the BackingMap. If this method is called on the ObjectMap and the `TTLType` is something other than `LAST_ACCESS_TIME` or `LAST_UPDATE_TIME`, an IllegalStateException is thrown.

Required permission: `MapPermission.INVALIDATE`

Parameters:

`ttl` - is the time-to-live value in seconds. The value must be >= -1. A value of 0 is used to indicate the cache entry can live "forever" and -1 to indicate to use default setting. Use of the constant `TTL_FOREVER` is recommended when "forever" is desired and the constant`USE_DEFAULT` is recommended when "use default" setting is desired.

Returns:

previous time-to-live value in seconds. The constant `TTL_FOREVER` and constant `USE_DEFAULT`can be used to determine if the previous TTL is one of the special values.

Throws:

[IllegalArgumentException](#) - if seconds argument is < -1.

[IllegalStateException](#) - if `BackingMap.getTtlEvictorType()` returns anything other than `TTLType.LAST_ACCESS_TIME` or `TTLType.LAST_UPDATE_TIME`.

[AccessControlException](#) - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

[TTL_FOREVER](#), [USE_DEFAULT](#), [BackingMap.setTimeToLive(int)](#), [TTLType.LAST_ACCESS_TIME](#), [TTLType.LAST_UPDATE_TIME](#)

---

update

```
void update(Object key,
            Object value)
            throws KeyNotFoundException,
                   ObjectGridException
```

Performs an explicit update of a given entry.

A `get` operation is not required prior to invoking the `update` method (unlike the `put` method). Also, an `update` invocation will never insert a new record. If a the map's `LockStrategy` is `LockStrategy.OPTIMISTIC` this method will implicitly get the entry so as to have the version value of the object for when this method was invoked. Whether or not a copy of the object is made when transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If a key cannot be found in the map during commit, a TransactionException will be thrown.

Specification details:

Required permission:	`MapPermission.WRITE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - Identifies the entry to be updated

`value` - The updated value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.

`KeyNotFoundException` - if the key cannot be found in the map

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`insert(Object, Object)`, `put(Object, Object)`, `CopyMode`, `LockStrategy.OPTIMISTIC`

---

insert

```
void insert(Object key,
            Object value)
            throws DuplicateKeyException,
                   ObjectGridException
```

Performs an explicit insert of a given entry.

The key must not exist before executing this method. Also, an `insert` invocation will never update an existing record. Whether or not a copy of the object is made when a transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If the key is already in the map, a TransactionException will be thrown during commit.

Specification details:

Required	`MapPermission.INSERT`

permissi on:	
Transact ion:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - Identifies the entry to be inserted

`value` - The value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.

`DuplicateKeyException` - if this entries already exists in the map

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`put(Object, Object)`, `update(Object, Object)`, `CopyMode`

---

getIndex

```
Object getIndex(String name)
                throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                       com.ibm.websphere.objectgrid.IndexNotReadyException
```

Returns a reference to the named index that can be used with this Map. This index cannot be shared between threads and works on the same rules as `Session`. The returned value should be cast to the right index interface such as `MapIndex`, `MapRangeIndex` or a custom index interface such as a geo spatial index.

Parameters:

`name` - The index name

Returns:

A reference to the index, it must be cast to the appropriate index interface.

Throws:

`IndexUndefinedException` - if the index is not defined on the BackingMap

`IndexNotReadyException` - if the index is a dynamic index and it is not ready

Since:

WAS XD 6.0.1

---

getIndex

```
Object getIndex(String name,
                boolean forUpdate)
                throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                       com.ibm.websphere.objectgrid.IndexNotReadyException
```

Returns a reference to the named index that can be used with this Map. This index cannot be shared between threads and works on the same rules as `Session`. The returned value should be cast to the right index interface such as `MapIndex`, `MapRangeIndex` or a custom index interface such as a geo spatial index.

Parameters:

`name` - The index name

`forUpdate` - if true, the returned index will always operate with forUpdate intent.

Returns:

A reference to the index, it must be cast to the appropriate index interface.

Throws:

`IndexUndefinedException` - if the index is not defined on the BackingMap

`IndexNotReadyException` - if the index is a dynamic index and it is not ready

Since:

WAS XD 6.1.0.1

---

flush

```
void flush()
        throws ObjectGridException
```

Pushes the current set of changes for the `ObjectMap` instance to the `Loader` without committing the changes. The changes are not propagated to the `BackingMap` either. This is useful for re-priming the `Loader`'s data without committing the current transaction and starting over.

Throws:

`ObjectGridException` - if an error occurs during processing

See Also:

`Session.flush()`

---

containsKey

```
boolean containsKey(Object key)
                    throws ObjectGridException
```

Returns `true` if this map contains a mapping for the specified key. ObjectGrid does not support null keys. If you configured the map to support `null` values, this method can be used to determine whether a key is contained in the map or not.

This API does not hold any locks when using pessimistic locking.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`
Pessimistic locks held:	None
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found.

Parameters:

`key` - key whose presence in this map is to be tested.

Returns:

`true` if this map contains a mapping for the specified key.

Throws:

`IllegalArgumentException` - if `null` key parameter is passed in

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

getJavaMap

```
Map getJavaMap()
```

Returns an implementation of `java.util.Map` that is backed by this `ObjectMap`.

The returned `java.util.Map` implementation can be cast to `com.ibm.websphere.objectgrid.JavaMap` to be able to use the rest of the ObjectGrid

programming model, but with `java.util.Map`'s use of `RuntimeException`s instead of checked `ObjectGridException`s.

Returns:

a `java.util.Map` backed by this `ObjectMap`

See Also:

`Map`, `JavaMap`

---

touch

```
void touch(Object key)
           throws ObjectGridException
```

Updates the last access time in the BackingMap without retrieving the value to the ObjectMap.

The last access time is updated during commit. If the key does not exist in the BackingMap, a TransactionException will be returned during commit processing.
Specification details:

Required permission:	None
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - key to be touched

Throws:

`IllegalArgumentException` - if key is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

setCopyMode

```
void setCopyMode(CopyMode copyMode,
                 Class valueInterface)
                 throws TransactionAlreadyActiveException,
                        ObjectGridException
```

Allows the CopyMode for the Map to be overridden on this map on this session only.

This method allows an application to use an optimal CopyMode TRANSACTION by TRANSACTION as its needs dictate. The CopyMode cannot be changed during a transaction. There must be no active transaction when this method is called.

Parameters:

`copyMode` - must be one of the final static variables defined in `CopyMode`. See `CopyMode` class for an explanation of each mode and how the valueInterface is used for `CopyMode.COPY_ON_WRITE` .
`valueInterface` - the value interface Class object. Specify null in version 7.1 and later.

Throws:

`IllegalArgumentException` - if copyMode is `null` or COPY_ON_WRITE CopyMode is specified and the required value interface parameter is `null`
`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

ObjectGridException - if an error occurs during processing

See Also:

BackingMap.setCopyMode(CopyMode, Class), CopyMode

---

clearCopyMode

```
void clearCopyMode()
                throws TransactionAlreadyActiveException
```

Resets the CopyMode back to the one in the BackingMap.

This method is used to reverse a previous setCopyMode method call for this ObjectMap. This method can only be called when no transaction is active on the associated session.

Throws:

TransactionAlreadyActiveException - if a transaction is active and this map has already been used in the transaction.

See Also:

setCopyMode(CopyMode, Class)

---

getNextKey

```
Object getNextKey(long timeout)
                  throws ObjectGridException
```

Retrieves a key off the map in first-in-first-out (FIFO) insert order.

The entry is locked by the session such that other calls to getNextKey will not return the same key. The key can be used to remove or manipulate the value although leaving the entry will result in the key remaining at the beginning of the queue. This order is optimized for performance and is not guaranteed especially across partitions or in highly concurrent environments.

The return value is a `SerializedKey` when OutputFormat.RAW is set for the keys. The default key output format for maps that are associated with a KeySerializerPlugin is OutputFormat.RAW. The SerializedKey allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.EXCLUSIVE`
Pessimistic read lock held:	Yes
Transaction:	Automatic or manual

Parameters:

`timeout` - The period of time in milliseconds to wait for an entry to become available on the queue.

Returns:

the next key

Throws:

ObjectGridException - if an error occurs during processing

AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

WAS XD 6.1

See Also:

QUEUE_TIMEOUT_INFINITE, QUEUE_TIMEOUT_NONE

---

getEntityMetadata

`com.ibm.websphere.projector.md.EntityMetadata` **`getEntityMetadata`**`()`

Retrieve the metadata for the entity associated with this map.

Returns:
the EntityMetadata if an entity is associated with this map or null if there is no entity associated with this map.
Since:
WAS XD 6.1

---

getMapType

`int` **`getMapType`**`()`

Returns the type of the underlying BackingMap.

The return value is equivalent to one of the constants declared on the BackingMap interface, `BackingMap.LOCAL`, `BackingMap.SERVER`, or `BackingMap.CLIENT`.

Returns:
the BackingMap type
Since:
WAS XD 6.1

---

getAgentManager

`com.ibm.websphere.objectgrid.datagrid.AgentManager` **`getAgentManager`**`()`

Returns the Agent Manager that allows DataGrid operations to be performed on the objects within this Map.

This method should only be called on a client ObjectGrid. If called on a non client ObjectGrid an `IllegalStateException` will be thrown

Returns:
AgentManager
Throws:
`IllegalStateException` - if this method is invoked on a non client ObjectGrid
Since:
WAS XD 6.1

---

setLockTimeout

`void` **`setLockTimeout`**`(int seconds)`

Overrides the BackingMap's lock timeout for this ObjectMap.

Establishes the number of seconds that any given fetch (get, getForUpdate, find, findForUpdate) of a cache entry will wait to get a lock. When the lock strategy is `LockStrategy.NONE`, no lock manager is used by this map. In this case, a call to this method does nothing.

Parameters:
`seconds` - is the lock timeout in seconds. The value must be >= -1. A value of -1 is

used to indicate to use the default setting. Use of the constant`USE_DEFAULT` is recommended when "use default" setting is desired. A value of 0 indicates that if a lock cannot be retrieved immediately to time out without waiting for any period of time for the lock to be released and made available.

Throws:

`IllegalArgumentException` - if seconds argument is less than -1 (USE_DEFAULT)

Since:

WAS XD 6.1

See Also:

`USE_DEFAULT`, `BackingMap.setLockTimeout(int)`, `BackingMap.setLockStrategy(LockStrategy)`, `LockStrategy.OPTIMISTIC`, `LockStrategy.PESSIMISTIC`

---

clear

```
void clear()
          throws ObjectGridException
```

Clear all keys from the Map.

This method is an automatic transaction call. The `Session.isTransactionActive()` must answer false prior to invoking this method.

Specification details:

Required permission:	`MapPermission.REMOVE`
Pessimistic read lock acquired:	Map exclusive lock is acquired.
Transaction:	Hybrid. Each shard is cleared in a separate transaction.
Cache tier:	Applies to all cache tiers except the Loader.

Throws:

`ObjectGridException` - if an error occurs during processing
`TransactionAlreadyActiveException` - if a transaction is already started.
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

WAS XD 6.1.0.3

---

lock

```
boolean lock(Object key,
             com.ibm.websphere.objectgrid.LockMode lockMode)
             throws ObjectGridException
```

Obtains a lock for the given key. **Specification details:**

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`, `LockMode.UPGRADABLE` or `LockMode.EXCLUSIVE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

Parameters:

`key` - the key to lock
`lockMode` - the lockMode to obtain for the given key

Returns:

true if the entry exists in the grid or Loader (if one is defined)

Throws:

IllegalArgumentException - if key is `null`
IllegalStateException - if this map is not using LockStrategy.PESSIMISTIC LockStategy
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

---

lockAll

```
List<Boolean> lockAll(List keyList,
                      com.ibm.websphere.objectgrid.LockMode lockMode)
                      throws ObjectGridException
```

Obtains locks for the given keys. **Specification details:**

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`, `LockMode.UPGRADABLE` or `LockMode.EXCLUSIVE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the keys are found and the appropriate locks are acquired.

Parameters:

`keyList` - the keys to lock
`lockMode` - the lockMode to obtain for the given keys

Returns:

a List of `Booleans` indicating whether the entries exists in the grid or Loader (if one is defined)

Throws:

IllegalStateException - if this map is not using LockStrategy.PESSIMISTIC LockStategy
IllegalArgumentException - if keyList is null or contains a `null` element.
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

---

upsert

```
void upsert(Object key,
            Object value)
            throws ObjectGridException
```

Puts the Object value into the cache at location represented by key.

The value will be pushed down to the BackingMap/Loader at commit time. The semantics of this method are that the Loader will receive a `LogElement.UPSERT` operation and the map will either do an insert or an update to cause the map to contain this updated value. Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

Specification details:

Required permissi on:	`MapPermission.WRITE`
Transact	Automatic or manual

ion:

Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - The entry to insert or update in the map
`value` - The value to insert or update in the map using the key

Throws:

`IllegalArgumentException` - if key is `null`, or if the map does not allow `null` values and value is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

See Also:

`CopyMode`

---

upsertAll

```
void upsertAll(LinkedHashMap map)
               throws ObjectGridException
```

Puts each of the Object value into the cache at location represented by key contained in the Map. The values will be pushed down to the BackingMap/Loader at commit time. The semantics of this method are that the Loader will receive a `LogElement.UPSERT` operation and the map will either do an insert or an update to cause the map to contain this updated value. Whether or not a copy of the objects is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

Specification details:

Required permission:	`MapPermission.WRITE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`map` - The key/values to be inserted or updated in the map. The type is LinkedHashMap so that the order can be controlled to avoid deadlocks.

Throws:

`IllegalArgumentException` - if map is `null` or contains a `null` key or if `null` values are not allowed and map contains a `null` value.
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

See Also:

`CopyMode`

---

getKeyOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getKeyOutputFormat`**`()`

Retrieves the data format for all data access APIs that return cache keys for the current session.

Returns:
the data output format or null if the default should be used.

Since:
8.6, XC10 2.5

---

setKeyOutputFormat

`void` **`setKeyOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`
`throws TransactionAlreadyActiveException`

Sets the data format for all data access APIs that return cache keys for the current session.

This method supports map configurations with a `KeyDataSerializer` plug-in defined, or with eXtreme Data Format enabled.

Parameters:
`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default defined on the parent `BackingMap`.

Throws:
`IllegalArgumentException` - thrown when the data format is not valid for the current configuration.
`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

Since:
8.6, XC10 2.5

See Also:
`BackingMap.getKeyOutputFormat()`

---

getValueOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getValueOutputFormat`**`()`

Retrieves the data format for all data access APIs that return cache values for the current session.

Returns:
the data output format or null if the default should be used.

Since:
8.6, XC10 2.5

---

setValueOutputFormat

`void` **`setValueOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`
`throws TransactionAlreadyActiveException`

Sets the data format for all data access APIs that return cache values for the current session.

This method is functionally equivalent to the `setCopyMode(CopyMode, Class)` with the `CopyMode.COPY_TO_BYTES` and `CopyMode.COPY_TO_BYTES_RAW` values when used with a `ValueDataSerializer` or eXtreme Data Format.

This method supports map configurations with a `ValueDataSerializer` plug-in defined, or with eXtreme Data Format enabled.

Parameters:

`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default defined on the parent `BackingMap`.

Throws:

`IllegalArgumentException` - thrown when the data format is not valid for the current configuration.

`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

Since:

8.6, XC10 2.5

---

setPutMode

```
void setPutMode(ObjectMap.PutMode putMode)
                throws TransactionAlreadyActiveException,
                       ObjectGridException
```

Allows the default operation for the `put(Object, Object)` and `putAll(Map)` methods to be changed, allowing the `put` to use the optimized `upsert(Object, Object)` and `upsertAll(LinkedHashMap)` implementations.

The PutMode cannot be changed during a transaction. There must be no active transaction when this method is called.

Parameters:

`putMode` - the mode in which the `put` methods operate.

Throws:

`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

`ObjectGridException` - if an error occurs during processing

Since:

8.6, XC10 2.5

---

getPutMode

`ObjectMap.PutMode` **`getPutMode`**`()`

Retrieve the current `PutMode` set for this ObjectMap instance.

Returns:

the current `PutMode`.

Since:

8.6, XC10 2.5

---

com.ibm.websphere.objectgrid

Enum ObjectMap.PutMode

java.lang.Object
  java.lang.Enum<ObjectMap.PutMode>
    com.ibm.websphere.objectgrid.ObjectMap.PutMode

All Implemented Interfaces:

Serializable, Comparable<ObjectMap.PutMode>

Enclosing interface:

ObjectMap

```
public static enum ObjectMap.PutMode
extends Enum<ObjectMap.PutMode>
```

Identifies the operation mode of the ObjectMap.put(Object, Object), ObjectMap.putAll(Map) , JavaMap.put(Object, Object) and JavaMap.putAll(Map) methods.

Since:

8.6, XC10 2.5

See Also:

ObjectMap.setPutMode(PutMode)

Enum Constant Summary

INSERTUPDATE **Deprecated.** *Deprecated in 8.6. Use the UPSERT enumeration.*
UPSERT The put API behaves like the upsert method.

Method Summary

static ObjectMap.PutMode	**valueOf**(String name) Returns the enum constant of this type with the specified name.

de	
static ObjectMap.PutMode[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

INSERTUPDATE

```
public static final ObjectMap.PutMode INSERTUPDATE
```

Deprecated. *Deprecated in 8.6. Use the UPSERT enumeration.*
The put API behaves like an insert or update.

UPSERT

```
public static final ObjectMap.PutMode UPSERT
```

The put API behaves like the upsert method.

Method Detail

values

```
public static ObjectMap.PutMode[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (ObjectMap.PutMode c : ObjectMap.PutMode.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

---

valueOf

public static ObjectMap.PutMode **valueOf**(String name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

name - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

IllegalArgumentException - if this enum type has no constant with the specified name

NullPointerException - if the argument is null

---

com.ibm.websphere.objectgrid

Class ObjectGridRuntimeException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        com.ibm.websphere.objectgrid.ObjectGridRuntimeException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

TransactionTimeoutException

```
public class ObjectGridRuntimeException
extends RuntimeException
implements IObjectGridException
```

This exception is the base class for all runtime exceptions thrown by the cache.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridRuntimeException() Constructs a new ObjectGridRuntimeException with `null` as its detail message.
ObjectGridRuntimeException(String message) Constructs a new ObjectGridRuntimeException with the specified detail message.
ObjectGridRuntimeException(String message, Throwable cause) Constructs a new ObjectGridRuntimeException with the specified detail message and cause.
ObjectGridRuntimeException(Throwable cause) Constructs a new ObjectGridRuntimeException with a specified cause.

Method Summary

Throwable	**getCause**() Returns the cause of this ObjectGridRuntimeException or `null` if the cause is nonexistent or unknown.

Throwable	**initCause**(Throwable cause) Initializes the *cause* of this ObjectGridRuntimeException to the specified value.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridRuntimeException

public **ObjectGridRuntimeException**()

Constructs a new ObjectGridRuntimeException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
initCause(Throwable)

ObjectGridRuntimeException

public **ObjectGridRuntimeException**(String message)

Constructs a new ObjectGridRuntimeException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
initCause(Throwable), Throwable.getMessage()

ObjectGridRuntimeException

public **ObjectGridRuntimeException**(Throwable cause)

Constructs a new ObjectGridRuntimeException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ObjectGridRuntimeExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

getCause()

ObjectGridRuntimeException

```
public ObjectGridRuntimeException(String message,
                                  Throwable cause)
```

Constructs a new ObjectGridRuntimeException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridRuntimeException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

getCause(), Throwable.getMessage()

Method Detail

getCause

```
public Throwable getCause()
```

Returns the cause of this ObjectGridRuntimeException or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this ObjectGridRuntimeException to get thrown.)

This implementation returns the cause that was supplied via one of the constructors requiring a `Throwable`, or that was set after creation with the `initCause(Throwable)` method. While it is typically unnecessary to override this method, a subclass can override it to return a cause set by some other means. This is appropriate for a "legacy chained throwable" that predates the addition of chained exceptions to `Throwable`. Note that it is *not* necessary to override any of the `PrintStackTrace` methods, all of which invoke the `getCause` method to determine the cause of an ObjectGridRuntimeException

Specified by:

`getCause` in interface `IObjectGridException`

Overrides:

`getCause` in class `Throwable`

Returns:

the cause of this ObjectGridRuntimeException or `null` if the cause is nonexistent or unknown.

See Also:

ObjectGridRuntimeException(String, Throwable), ObjectGridRuntimeException(Throwable), initCause(Throwable)

---

initCause

```
public Throwable initCause(Throwable cause)
```

Initializes the *cause* of this ObjectGridRuntimeException to the specified value. (The cause is the throwable that caused this ObjectGridRuntimeException to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the ObjectGridRuntimeException. If this

ObjectGridRuntimeException was created with `ObjectGridRuntimeException(Throwable)` or `ObjectGridRuntimeException(String,Throwable)`, this method cannot be called even once.

Specified by:

`initCause` in interface `IObjectGridException`

Overrides:

`initCause` in class `Throwable`

Parameters:

`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)

Returns:

a reference to this `ObjectGridRuntimeException` instance.

Throws:

`IllegalArgumentException` - if `cause` is this ObjectGridRuntimeException. (An ObjectGridRuntimeException cannot be its own cause.)

`IllegalStateException` - if this ObjectGridRuntimeException was created with `ObjectGridRuntimeException(Throwable)` or `ObjectGridRuntimeException(String,Throwable)`, or this method has already been called on this ObjectGridRuntimeException.

See Also:

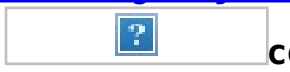
`ObjectGridRuntimeException(String, Throwable)`, `ObjectGridRuntimeException(Throwable)`, `getCause()`

---

com.ibm.websphere.objectgrid

Class ObjectGridManagerFactory

java.lang.Object



com.ibm.websphere.objectgrid.ObjectGridManagerFactory

```
public final class ObjectGridManagerFactory
extends Object
```

This factory class is a high level helper class to get `ObjectGridManager` instances.

Since:

WAS XD 6.0, XC10

Constructor Summary

ObjectGridManagerFactory()

Method Summary

static ObjectGridManager	**getObjectGridManager**() Returns the `ObjectGridManager` singleton.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridManagerFactory

public **ObjectGridManagerFactory**()

Method Detail

getObjectGridManager

public static final ObjectGridManager **getObjectGridManager**()

Returns the `ObjectGridManager` singleton.

Returns:
The ObjectGridManager singleton
See Also:
ObjectGridManager

---

com.ibm.websphere.objectgrid

Interface ObjectGridManager

public interface **ObjectGridManager**

ObjectGridManager is responsible for creating or retrieving local ObjectGrid instances and connecting to distributed ObjectGrid servers. Use the `ObjectGridManagerFactory` to retrieve an ObjectGridManager.

Use the `createObjectGrid` methods to create a local, in-memory ObjectGrid instance. The `createObjectGrid` methods give you the choice of caching the created ObjectGrid instance. If you choose to cache the instance, you cannot create an ObjectGrid with the same name unless you remove the previously created ObjectGrid using the `removeObjectGrid(String)` method. A cached `ObjectGrid` instance can later be retrieved using the `getObjectGrid(String)` method.

An example of creating a local in-memory ObjectGrid programmatically:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid grid = ogMgr.createObjectGrid("LocalBookStoreGrid");
grid.defineMap("Orders");
grid.defineMap("Books");
grid.initialize();
...
grid.destroy();
```

An example of creating a local in-memory ObjectGrid using an ObjectGrid descriptor XML file:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
URL objectgridXML  = Thread.currentThread().getContextClassLoader().getResource("configs/objectgr
id.xml");
ObjectGrid grid = ogMgr.createObjectGrid("LocalBookStoreGrid", objectgridXML);
grid.initialize();
...
ogMgr.destroy();
```

Use the `connect` methods to connect to a distributed ObjectGrid. The connect methods return a ClientClusterContext that can then be passed to one of the `getObjectGrid` methods, which will in turn retrieve a client ObjectGrid instance.

An example to connect to a dynamic, distributed ObjectGrid using a catalog server cluster:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ClientClusterContext ccc = ogMgr.connect("catserver1:2809,catserver2:2809", null, null);
ObjectGrid grid = ogMgr.getObjectGrid(ccc, "BookStoreGrid");
...
ogMgr.disconnect(ccc);
```

An example to connect to an embedded ObjectGrid server (a server running in the current process):

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ClientClusterContext ccc = ogMgr.connect((ClientSecurityConfiguration) null, (URL) null);
ObjectGrid grid = ogMgr.getObjectGrid(ccc, "BookStoreGrid");
...
```

```
ogMgr.disconnect(ccc);
```

This interface also allows ObjectGrid trace to be disabled completely for performance improvements especially on a processor with a smaller L2 cache.

Since:
WAS XD 6.0, XC10

Method Summary

ClientClusterContext	**connect**(ClientSecurityConfiguration securityProps, URL overRideObjectGridXml) Use this method to connect to an embedded ObjectGrid server.
ClientClusterContext	**connect**(String catalogServerEndpoints, ClientSecurityConfiguration securityProps, URL overRideObjectGridXml) Connects a client process to a remote ObjectGrid catalog server or catalog server cluster (a dynamic ObjectGrid deployment topology).
ObjectGrid	**createObjectGrid**() Creates a local, in-memory instance of an `ObjectGrid`.
ObjectGr	**createObjectGrid**(String objectGridName) Creates and caches a local, in-memory ObjectGrid instance with the specified name.

Object	**createObjectGrid**(String objectGridName, boolean cacheInstance) Creates a local, in-memory instance of an ObjectGrid with the specified name.
Object	**createObjectGrid**(String objectGridName, URL xmlFile) Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.
Object	**createObjectGrid**(String objectGridName, URL xmlFile, boolean cacheInstance) Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.
Object	**createObjectGrid**(String objectGridName, URL xmlFile, boolean enableXmlValidation, boolean cacheInstance) **Deprecated.** *Deprecated in version 8.6. XML validation is always enabled. Use instead.*
List	**createObjectGrids**(URL xmlFile) Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.
List	**createObjectGrids**(URL xmlFile, boolean cacheInstances) Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.
List	**createObjectGrids**(URL xmlFile, boolean enableXmlValidation, boolean cacheInstances) **Deprecated.** *Deprecated in version 8.6. XML validation is always enabled. Use createObjectGrids(URL, boolean) instead.*
boolean	**disconnect**(ClientClusterContext context) Disconnects a client process from an ObjectGrid server
Ca	

CatalogDomainManager	**getCatalogDomainManager**() Retrieve the CatalogDomainManager for this ObjectGridManager.
ObjectGrid	**getObjectGrid**(ClientClusterContext context, String objectGridName) Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.
ObjectGrid	**getObjectGrid**(ClientClusterContext context, String objectGridName, ObjectGridConfiguration overrideOGConfig) Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.
ObjectGrid	**getObjectGrid**(String objectGridName) Returns a cached local, in-memory `ObjectGrid` instance by name.
List	**getObjectGrids**() Gets a `List` of the local, in-memory `ObjectGrid` instances that have been previously cached.
Map	**getOverrideObjectGridConfigurations**() **Deprecated.** *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*
String	**getTraceSpecification**() Retrieves the trace specification for the current JVM.
void	**putOverrideObjectGridConfigurations**(String clusterName, List objectGridConfigList) **Deprecated.** *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

void	**removeObjectGrid**(String objectGridName) Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances.
void	**removeObjectGrid**(String objectGridName, boolean destroy) Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances and optionally destroys its associated resources.
void	**setOverrideObjectGridConfigurations**(Map overrideMap) **Deprecated.** *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*
void	**setTraceEnabled**(boolean enabledFlag) Enables or disables ObjectGrid trace for the JVM.
void	**setTraceFileName**(String traceFileName) Sets the trace output to go to a file instead of System.out.
void	**setTraceSpecification**(String traceSpec) Set the trace specification for the current JVM.

Method Detail

createObjectGrid

ObjectGrid **createObjectGrid**()
throws ObjectGridException

Creates a local, in-memory instance of an ObjectGrid.

The created ObjectGrid returned by this method is assigned a unique name is **not** cached by the ObjectGridManager. Use the ObjectGrid.setName(String) method to change the ObjectGrid name.

Returns:
an instance of ObjectGrid with a unique name assigned
Throws:
ObjectGridException - if any error occurs during the ObjectGrid creation
See Also:
ObjectGrid, ObjectGrid.setName(String)

---

createObjectGrid

ObjectGrid **createObjectGrid**(String objectGridName,
boolean cacheInstance)
throws ObjectGridException

Creates a local, in-memory instance of an ObjectGrid with the specified name.

The instance of ObjectGrid created can optionally be cached. If an ObjectGrid with the specified name was previously created and cached, an ObjectGridException will be thrown.

Parameters:

`objectGridName` - the name of the ObjectGrid to be created. Must not be `null`.
`cacheInstance` - true, if the ObjectGrid instance should be cached

Returns:

an ObjectGrid instance with the specified name.

Throws:

`IllegalArgumentException` - if objectGridName is `null`
`ObjectGridException` - if an ObjectGrid with this name has already been cached or any error occurs during the ObjectGrid creation.

See Also:

`ObjectGrid`

---

createObjectGrid

```
ObjectGrid createObjectGrid(String objectGridName)
                            throws ObjectGridException
```

Creates and caches a local, in-memory ObjectGrid instance with the specified name.

The ObjectGrid instance created by this method will be cached. Invoking this method is equivalent to invoke `createObjectGrid(String, true)`

Parameters:

`objectGridName` - the Name of the ObjectGrid instance to be created. Must not be `null`.

Returns:

an ObjectGrid instance with the specified name

Throws:

`IllegalArgumentException` - if objectGridName is `null`
`ObjectGridException` - if an ObjectGrid with this name has already been cached, or any error occurs during the ObjectGrid creation

See Also:

`createObjectGrid(String, boolean)`, `ObjectGrid`

---

createObjectGrid

```
@Deprecated
ObjectGrid createObjectGrid(String objectGridName,
                                    URL xmlFile,
                                    boolean enableXmlValidation,
                                    boolean cacheInstance)
                            throws ObjectGridException
```

Deprecated. *Deprecated in version 8.6. XML validation is always enabled. Use `instead.`*

Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.

An ObjectGrid instance is created for the ObjectGrid configuration in the XML file corresponding to the specified ObjectGrid name. If the specified ObjectGrid name cannot be found in the XML file, an exception will be thrown.

This returned ObjectGrid instance optionally can be cached.

If the URL is null, it will be simply ignored. In this case, this method behaves the same as the `createObjectGrid(String, boolean)`.

Parameters:

`objectGridName` - the Name of the ObjectGrid instance to be returned. Must not be `null`.
`xmlFile` - a URL to a well formed xml file based on the ObjectGrid schema.
`enableXmlValidation` - if true the XML is validated
`cacheInstance` - a boolean value indicating whether the returned ObjectGrid instance

defined in the XML will be cached or not. If true, the instance will be cached.

Returns:

an ObjectGrid instance

Throws:

`IllegalArgumentException` - if objectGridName is `null`

`ObjectGridException` - if an ObjectGrid with the same name has been previously cached, an ObjectGrid configuration with the specified name can be found in the xml file, or any other error occurs during ObjectGrid creation.

See Also:

`createObjectGrid(String, boolean)`, `ObjectGrid`

---

createObjectGrid

```
ObjectGrid createObjectGrid(String objectGridName,
                            URL xmlFile,
                            boolean cacheInstance)
                            throws ObjectGridException
```

Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.

An ObjectGrid instance is created for the ObjectGrid configuration in the XML file corresponding to the specified ObjectGrid name. If the specified ObjectGrid name cannot be found in the XML file, an exception will be thrown.

This returned ObjectGrid instance optionally can be cached.

If the URL is null, it will be simply ignored. In this case, this method behaves the same as the `createObjectGrid(String, boolean)`.

Parameters:

`objectGridName` - the Name of the ObjectGrid instance to be returned. Must not be `null`.

`xmlFile` - a URL to a well formed xml file based on the ObjectGrid schema.

`cacheInstance` - a boolean value indicating whether the returned ObjectGrid instance defined in the XML will be cached or not. If true, the instance will be cached.

Returns:

an ObjectGrid instance

Throws:

`IllegalArgumentException` - if objectGridName is `null`

`ObjectGridException` - if an ObjectGrid with the same name has been previously cached, an ObjectGrid configuration with the specified name can be found in the xml file, or any other error occurs during ObjectGrid creation.

Since:

8.6, XC10 2.5

See Also:

`createObjectGrid(String, boolean)`, `ObjectGrid`

---

createObjectGrids

```
List createObjectGrids(URL xmlFile,
                       boolean enableXmlValidation,
                       boolean cacheInstances)
                       throws ObjectGridException
```

Deprecated. *Deprecated in version 8.6. XML validation is always enabled. Use `createObjectGrids(URL, boolean)` instead.*

Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.

The returned ObjecGrid instances can be cached. An ObjectGridException will be thrown when attempting to cache a newly created ObjectGrid that has the same name as an ObjectGrid that has already been cached.

Parameters:

`xmlFile` - the file that defines an ObjectGrid or multiple ObjectGrids

`cacheInstances` - set to true to cache all ObjectGrid instances created based on the file

Returns:

a list of ObjectGrid instances

Throws:

`ObjectGridException` - if attempting to create and cache an ObjectGrid with the same name as an ObjectGrid that has already been cached, or any other error occurs during ObjectGrid creation

See Also:

`ObjectGrid`

---

createObjectGrids

```
List createObjectGrids(URL xmlFile,
                       boolean cacheInstances)
                       throws ObjectGridException
```

Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.

The returned ObjecGrid instances can be cached. An ObjectGridException will be thrown when attempting to cache a newly created ObjectGrid that has the same name as an ObjectGrid that has already been cached.

Parameters:

`xmlFile` - the file that defines an ObjectGrid or multiple ObjectGrids

`cacheInstances` - set to true to cache all ObjectGrid instances created based on the file

Returns:

a list of ObjectGrid instances

Throws:

`ObjectGridException` - if attempting to create and cache an ObjectGrid with the same name as an ObjectGrid that has already been cached, or any other error occurs during ObjectGrid creation

Since:

8.6, XC10 2.5

See Also:

`ObjectGrid`

---

createObjectGrids

```
List createObjectGrids(URL xmlFile)
                       throws ObjectGridException
```

Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.

The XML file will be validated against the schema and each ObjectGrid instance that is created will be cached. An ObjectGridException will be thrown when attempting to cache a newly created ObjectGrid that has the same name as an ObjectGrid that has already been cached. Using this method is equivalent to calling the `createObjectGrids(URL, true, true)` method.

Parameters:

`xmlFile` - The XML file to process. ObjectGrid(s) will be created based on the configurations what is in the file.

Returns:

A list of ObjectGrid instances that have been created.

Throws:

`ObjectGridException` - if attempting to create and cache an ObjectGrid with the same name as an ObjectGrid that has already been cached, or any other error occurs during ObjectGrid creation

See Also:

`createObjectGrids(URL, boolean, boolean)`, `ObjectGrid`

---

createObjectGrid

```
ObjectGrid createObjectGrid(String objectGridName,
                            URL xmlFile)
                     throws ObjectGridException
```

Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.

If there is no ObjectGrid with this name defined in the XML file, an ObjectGridException will be thrown. The XML file will be validated against the schema and the ObjectGrid instance created will be cached. Using this method is equivalent to calling the `createObjectGrid(String, URL, true, true)` method.

Parameters:

`objectGridName` - name of the ObjectGrid to create. This ObjectGrid should be defined in the XML file. Must not be `null`.

`xmlFile` - the XML file to process

Returns:

A newly created ObjectGrid

Throws:

`IllegalArgumentException` - if objectGridName is `null`

`ObjectGridException` - if an ObjectGrid with the same name has been previously cached, an ObjectGrid configuration with the specified name can be found in the xml file, or any other error occurs during ObjectGrid creation.

See Also:

`createObjectGrid(String, URL, boolean, boolean)`, `ObjectGrid`

---

removeObjectGrid

```
void removeObjectGrid(String objectGridName)
                      throws ObjectGridException
```

Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances.

Invoking this method is equivalent to calling `removeObjectGrid(String, false)`

Parameters:

`objectGridName` - the name of the ObjectGrid instance to remove from the cache

Throws:

`ObjectGridException` - if an ObjectGrid with the objectGridName was not found in the cache

See Also:

`removeObjectGrid(String, boolean)`

---

removeObjectGrid

```
void removeObjectGrid(String objectGridName,
                      boolean destroy)
                      throws ObjectGridException
```

Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances and optionally destroys its associated resources.

Parameters:

`objectGridName` - the name of the ObjectGrid instance to remove from the cache
`destroy` - if true, destroy the objectgrid instance and its associated resources

Throws:

`ObjectGridException` - if an ObjectGrid with the objectGridName was not found in the cache

See Also:

`ObjectGrid.destroy()`

---

getObjectGrids

`List` **`getObjectGrids`**`()`

Gets a `List` of the local, in-memory `ObjectGrid` instances that have been previously cached.

This method returns `null` if no ObjectGrid instances have been cached.

Returns:

a list of ObjectGrid instances that have been previously cached or `null` if there are no cached ObjectGrid instances

---

getObjectGrid

`ObjectGrid` **`getObjectGrid`**`(``String` `objectGridName)

Returns a cached local, in-memory `ObjectGrid` instance by name.

This method returns `null` if no `ObjectGrid` with the specified name is currently cached.

Parameters:

`objectGridName` - the cached objectgrid name.

Returns:

a cached ObjectGrid which currently exists.

---

setTraceSpecification

`void` **`setTraceSpecification`**`(``String` `traceSpec)

Set the trace specification for the current JVM.

This operation is a replace operation, not an append operation. The specification should be of the form:

```
 TraceString := <ComponentString>(:<ComponentString>)* ComponentString :=  <ComponentName>=<t
ype>=<state>(,<type>=<state>)*
 ComponentName := a java String state := [enabled|disabled] type := [all|debug|event|entryExi
t]
```

For example, ObjectGrid=all=enabled

Parameters:

`traceSpec` - the new trace specification

---

getTraceSpecification

`String` **`getTraceSpecification`**`()`

Retrieves the trace specification for the current JVM.

Since:
7.1
See Also:
setTraceSpecification(String)

---

setTraceFileName

```
void setTraceFileName(String traceFileName)
```

Sets the trace output to go to a file instead of `System.out`.
The supplied file name can be relative to the working directory or a fully-qualified file name.

Parameters:
`traceFileName` - Name of trace file

---

setTraceEnabled

```
void setTraceEnabled(boolean enabledFlag)
```

Enables or disables ObjectGrid trace for the JVM.

Disabling trace improves the performance when ObjectGrid runs on processors whose L2 caches are not large enough to contain the trace enabled code paths. If this is set to false, ObjectGrid trace is suppressed even if it is enabled using `setTraceSpecification(String)`. By default ObjectGrid trace is enabled.

Parameters:
`enabledFlag` - true to enable trace
Since:
WAS XD 6.0.1
See Also:
setTraceSpecification(String)

---

connect

```
ClientClusterContext connect(ClientSecurityConfiguration securityProps,
                             URL overRideObjectGridXml)
                             throws ConnectException
```

Use this method to connect to an embedded ObjectGrid server. An embedded ObjectGrid server is typically started in an application server process such as IBM WebSphere Application Server. This method allows connecting to the in-memory ObjectGrid server instance without specifying connection information.

This method can be used to connect to both dynamic and static ObjectGrid server deployments.

Parameters:
`securityProps` - client security configuration. The value can be `null` if not running in secure mode.
`overRideObjectGridXml` - This parameter can be `null`. If it is not `null`, the client side configuration of the ObjectGrid plugins are overridden with the ObjectGrid configuration using this `URL`. If this parameter is `null`, client side ObjectGrid plugins can be overridden by providing an overrideMap to `setOverrideObjectGridConfigurations(Map)` or by calling `putOverrideObjectGridConfigurations(String, List)`.

In cases where this parameter is not null, and overriding configuration objects have been provided to the ObjectGridManager by `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)`, the configuration based on the XML file will be used to override ObjectGrid settings. Overriding objects provided to `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)` will be ignored.

Not all plugins can be overridden. For details please see the ObjectGrid documentation.

Returns:

a ClientClusterContext representing the cluster ObjectGrid configuration to which the client is connected.

Throws:

`ConnectException` - if any error occurs connecting to the server

Since:

WAS XD 6.0.1

See Also:

`ClientClusterContext`, `ClientSecurityConfiguration`

---

connect

```
ClientClusterContext connect(String catalogServerEndpoints,
                             ClientSecurityConfiguration securityProps,
                             URL overRideObjectGridXml)
                             throws ConnectException
```

Connects a client process to a remote ObjectGrid catalog server or catalog server cluster (a dynamic ObjectGrid deployment topology). The result ClientClusterContext can then be used to get any ObjectGrid reference managed by that catalog server cluster.

Parameters:

`catalogServerEndpoints` - A concatenated list of host/port pairs belonging to the catalog servers in the form "host:port<,host:port>". This list can be arbitrarily long and is used for bootstrapping only. The first viable address will be used.
`securityProps` - This parameter may be null if the client does not wish to establish a secure connection with the server.
`overRideObjectGridXml` - This parameter can be `null`. If it is not `null`, the client side configuration of the ObjectGrid plugins are overridden with the ObjectGrid configuration using this `URL`. If this parameter is `null`, client side ObjectGrid plugins can be overridden by providing an overrideMap to `setOverrideObjectGridConfigurations(Map)` or by calling `putOverrideObjectGridConfigurations(String, List)`.

In cases where this parameter is not null, and overriding configuration objects have been provided to the ObjectGridManager by `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)`, the configuration based on the XML file will be used to override ObjectGrid settings. Overriding objects provided to `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)` will be ignored.

Not all plugins can be overridden. For details please see the ObjectGrid documentation.

Returns:

a ClientClusterContext representing the cluster ObjectGrid configuration to which the client is connected.

Throws:

`ConnectException` - If there is a problem connecting to the addresses given.

---

disconnect

```
boolean disconnect(ClientClusterContext context)
```

Disconnects a client process from an ObjectGrid server

Parameters:

`context` - the `ClientClusterContext` object returned from a previous call to one of the `connect` methods The caller must guarantee this parameter is not `null`.

Returns:

`true` if the disconnect was successful, `false` if the supplied context was not connected

Throws:

`IllegalArgumentException` - if the ClientClusterContext is null

Since:

WAS XD 6.0.1

See Also:

`ClientClusterContext`

---

getObjectGrid

```
ObjectGrid getObjectGrid(ClientClusterContext context,
                         String objectGridName)
```

Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.

This method is equivalent to calling `getObjectGrid(context, objectGridName, null)`

Parameters:

`context` - the `ClientClusterContext` object returned from a previous call to one of the `connect` methods The caller must guarantee this parameter is not `null`.

`objectGridName` - the name of the requested client ObjectGrid

Returns:

a client ObjectGrid instance corresponding to a remote ObjectGrid

Throws:

`IllegalArgumentException` - if either provided parameter is null

`ObjectGridRuntimeException` - is the `ObjectGrid` with the specified name is not hosted in any eXtreme Scale servers managed by the catalog server

Since:

WAS XD 6.0.1

See Also:

`ClientClusterContext`, `ObjectGrid`, `getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)`

---

getObjectGrid

```
ObjectGrid getObjectGrid(ClientClusterContext context,
                         String objectGridName,
                         ObjectGridConfiguration overrideOGConfig)
```

Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.

This method replaces the get/set/putOverrideObjectGridConfigurations methods. Those methods had thread safety issues. In addition they were global in nature so we end up having configuration override happen for all client connections unless it was managed correctly. If ClientClusterContext was used previously to get an ObjectGrid for the given name, the same ObjectGrid instance is returned even if the `overrideOGConfig` parameter is different.

Parameters:

`context` - the `ClientClusterContext` object returned from a previous call to one of the `connect` methods The caller must guarantee this parameter is not `null`.
`objectGridName` - the name of the requested client ObjectGrid
`overrideOGConfig` - This parameter can be `null`. If it is not `null`, the client side configuration of the ObjectGrid plugins are overridden with the `ObjectGridConfiguration` provided. The provided override configuration takes precedence over any other provided override configuration for the requested ObjectGrid name provided by the `connect`, `putOverrideObjectGridConfigurations`, and `putOverrideObjectGridConfigurations` methods.

Not all plugins can be overridden. For details please see the ObjectGrid documentation.

Returns:

a client ObjectGrid instance corresponding to a remote ObjectGrid

Since:

7.1.1

See Also:

`ObjectGridConfiguration`, `ObjectGridConfigFactory`

---

setOverrideObjectGridConfigurations

`void` **`setOverrideObjectGridConfigurations`**`(Map overrideMap)`

Deprecated. *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

Override ObjectGrid settings on client side ObjectGrids by passing in a Map where each key corresponds to a cluster name or domain name and each value is a `List` of `ObjectGridConfiguration` objects to be overridden.

Client side configuration of ObjectGrid and BackingMap plugins are overridden using the ObjectGridConfiguration values provided in the List. To override a Plugin, each `ObjectGridConfiguration` object must have a name that matches the name of the ObjectGrid to be overridden. `BackingMapConfiguration` objects must have the same name as a BackingMap and be associated with the proper`ObjectGridConfiguration`.

Not all plugins can be overridden. ObjectGrid plugins that can be overridden on the client side are TransactionCallback and ObjectGridEventListener. BackingMap plugins that can be overridden on the client side are Evictor and MapEventListener. Settings for the builtin Evictor can also be altered on the BackingMap. These settings include numberOfBuckets, timeToLive, and ttlEvictorType.

Parameters:

`overrideMap` - a Map that will be used to override ObjectGrid settings on the client side. To override client side settings, each key of the Map must be a String with a value that corresponds to a cluster name or domain name. Each value of the overrideMap must be a `java.util.List`. The List elements must be `ObjectGridConfiguration` objects. Each call to a connect method with a clusterName that matches a key in the overrideMap will result in the client side settings being overridden using the List of ObjectGridConfiguration objects provided in the key's corresponding value. Pass in `null` to clear an overrideMap that was previously set and thereby remove any overriding settings from future connect calls.

Throws:

`IllegalArgumentException` - if any keys or values are null or if keys or values are of the wrong type

Since:

WAS XD 6.0.1.2

See Also:

`connect(String, ClientSecurityConfiguration, URL)`, `connect(ClientSecurityConfiguration,`

URL), ObjectGridConfiguration, BackingMapConfiguration, getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)

---

putOverrideObjectGridConfigurations

```
void putOverrideObjectGridConfigurations(String clusterName,
                                         List objectGridConfigList)
```

Deprecated. *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

Put an entry into the Map that is used to override client side ObjectGrid and BackingMap plugins.

Parameters:

`clusterName` - to be used as a key in the Map used to override client side ObjectGrid plugins. If a connect method is called with a matching clusterName, the client side ObjectGrid and BackingMap plugins can be overridden using the objectGridConfigList. In the dynamic environment, use the domain name to override ObjectGrid settings.

`objectGridConfigList` - a List of `ObjectGridConfiguration` objects that will be used to override client side ObjectGrid settings if a connect method is called with a cluster name that matches the clusterName on this method

Throws:

`IllegalArgumentException` - if the clusterName or objectGridConfigList is null

See Also:

getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)

---

getOverrideObjectGridConfigurations

```
Map getOverrideObjectGridConfigurations()
```

Deprecated. *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

Get the Map that is used to override client side ObjectGrid and BackingMap plugins.

Returns:

the Map that was set by the call to `setOverrideObjectGridConfigurations`. The Map may also have entries that were put there using the `putOverrideObjectGridConfigurations` method.

See Also:

getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)

---

getCatalogDomainManager

```
CatalogDomainManager getCatalogDomainManager()
```

Retrieve the CatalogDomainManager for this ObjectGridManager.

Returns:

the CatalogDomainManager, if available. Returns null if a CatalogDomainManager is not supported in the current runtime environment.

Since:

8.5

---

com.ibm.websphere.objectgrid

Class ObjectGridException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

CacheEntryException, ConnectException, ContinuousQueryException, DuplicateKeyException, KeyNotFoundException, LoaderException, LockException, NoActiveTransactionException, ObjectGridConfigurationException, ObjectGridSecurityException, ReadOnlyException, TransactionCallbackException, TransactionException, UndefinedMapException

---

```
public class ObjectGridException
extends Exception
implements IObjectGridException
```

Base exception class for all checked exceptions thrown by the ObjectGrid product.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridException() Constructs a new ObjectGridException with `null` as its detail message.
ObjectGridException(String message) Constructs a new ObjectGridException with the specified detail message.
ObjectGridException(String message, Throwable cause) Constructs a new ObjectGridException with the specified detail message and cause.
ObjectGridException(Throwable cause) Constructs a new ObjectGridException with a specified cause.

Method Summary

Throwable	**getCause**() Returns the cause of this ObjectGridException or `null` if the cause is nonexistent or unknown.

le	
Throwable	**initCause**(Throwable cause) Initializes the *cause* of this ObjectGridException to the specified value.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridException

public **ObjectGridException**()

Constructs a new ObjectGridException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
initCause(Throwable)

ObjectGridException

public **ObjectGridException**(String message)

Constructs a new ObjectGridException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
initCause(Throwable), Throwable.getMessage()

ObjectGridException

public **ObjectGridException**(Throwable cause)

Constructs a new ObjectGridException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ObjectGridExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:
getCause()

ObjectGridException

```
public ObjectGridException(String message,
                           Throwable cause)
```

Constructs a new ObjectGridException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
getCause(), Throwable.getMessage()

Method Detail

getCause

```
public Throwable getCause()
```

Returns the cause of this ObjectGridException or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this ObjectGridException to get thrown.)

This implementation returns the cause that was supplied via one of the constructors requiring a `Throwable`, or that was set after creation with the `initCause(Throwable)` method. While it is typically unnecessary to override this method, a subclass can override it to return a cause set by some other means. This is appropriate for a "legacy chained throwable" that predates the addition of chained exceptions to `Throwable`. Note that it is *not* necessary to override any of the `PrintStackTrace` methods, all of which invoke the `getCause` method to determine the cause of an ObjectGridException

Specified by:
`getCause` in interface `IObjectGridException`
Overrides:
`getCause` in class `Throwable`

Returns:
the cause of this ObjectGridException or `null` if the cause is nonexistent or unknown.
See Also:
ObjectGridException(String, Throwable), ObjectGridException(Throwable), initCause(Throwable)

---

initCause

```
public Throwable initCause(Throwable cause)
```

Initializes the *cause* of this ObjectGridException to the specified value. (The cause is the throwable that caused this ObjectGridException to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the ObjectGridException. If this

ObjectGridException was created with `ObjectGridException(Throwable)` or `ObjectGridException(String,Throwable)`, this method cannot be called even once.

Specified by:

`initCause` in interface `IObjectGridException`

Overrides:

`initCause` in class `Throwable`

Parameters:

`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)

Returns:

a reference to this `ObjectGridException` instance.

Throws:

`IllegalArgumentException` - if `cause` is this ObjectGridException. (An ObjectGridException cannot be its own cause.)

`IllegalStateException` - if this ObjectGridException was created with `ObjectGridException(Throwable)` or `ObjectGridException(String,Throwable)`, or this method has already been called on this ObjectGridException.

See Also:

`ObjectGridException(String, Throwable)`, `ObjectGridException(Throwable)`, `getCause()`

---

com.ibm.websphere.objectgrid

Interface ObjectGrid

public interface **ObjectGrid**

This object is used for creating sessions to the ObjectGrid. It is the central core of the ObjectGrid framework. Besides creating `Sessions`, it is also responsible for defining `BackingMaps`, setting a `TransactionCallback`, adding event listeners, and managing the security settings.

Since:

WAS XD 6.0, XC10

Field Summary

static int	**CLIENT** Indicates the ObjectGrid is a client-side distributed ObjectGrid.
static int	**DEFAULT_TX_TIMEOUT_VALUE** The default transaction time out value of 10 minutes if no transaction time out value is set.
static int	**LOCAL** Indicates the ObjectGrid is a local ObjectGrid.
static int	**SERVER** Indicates the ObjectGrid is a server-side distributed ObjectGrid.

Method Summary

void	**addEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener) Adds an EventListener.
void	**addEventListener**(ObjectGridEventListener listener) **Deprecated.** *This method is deprecated in version 7.1.1, use the addEventListener(EventListener) method.*
BackingMap	**createMap**(String name) Creates a BackingMap, but does not associate it with this ObjectGrid.
BackingMap	**defineMap**(String name) Defines a BackingMap that will be used by the application.
void	**destroy**() Destroys this instance.
com.ibm.websphere.objectgrid.revision	**getCollisionArbiter**() Retrieves the CollisionArbiter that this grid is using to resolve revision collisions.

CollisionArbiter	
List	**getEventListeners**() Returns the current list of EventListeners.
List	**getListOfMapNames**() Gets the list of map names currently defined for this ObjectGrid instance.
BackingMap	**getMap**(String name) Returns a BackingMap previously configured by calling the defineMap(String) or setMaps(List) method.
String	**getName**() Gets the name of this ObjectGrid.
int	**getObjectGridType**() Returns the type of ObjectGrid.
Session	**getSession**() Gets a Session object that can be used by a single thread at a time.
Session	**getSession**(CredentialGenerator credGen) Get a session using a CredentialGenerator.
Session	**getSession**(Subject subject) Allows the use of a specific Subject rather than use the SubjectSource configured on the ObjectGrid to get a Session.

com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener.State	**getState**() Retrieve the current life cycle state of this ObjectGrid.
Trans	

ActionCallback	**getTransactionCallback**() Retrieves the TransactionCallback object.
int	**getTxIsolation**() Retrieves the default transaction isolation level.
int	**getTxTimeout**() Gets transaction timeout setting for this ObjectGrid instance.
void	**initialize**() Begins the bootstrapping of the ObjectGrid and Session instances.
boolean	**isSecurityEnabled**() Checks whether security is enabled on this ObjectGrid or not.
void	**registerEntities**(Class[] entities) Register one or more entities based on the class metadata.
void	**registerEntities**(URL entityXML) Registers one ore more entities from an entity XML file.
void	**removeEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener) Removes an EventListener.
void	**removeEventListener**(ObjectGridEventListener listener) **Deprecated.** *This method is deprecated in version 7.1.1, use the removeEventListener(EventListener) method.*
int	**reserveSlot**(String containerName) Allows plugins on this ObjectGrid to reserve slots for use to store context data.
void	**setAccessByCreatorOnlyMode**(int accessByCreatorOnlyMode) Set the "access by creator only" mode.
void	**setAuthorizationMechanism**(int authMechanism) Sets the authorization mechanism.
v	**setCollisionArbiter**(com.ibm.websphere.objectgrid.revision.CollisionArbiter arbiter)

void	Sets the CollisionArbiter that is responsible for arbitration of revision conflicts.
void	**setEventListeners**(List listeners) **Deprecated.** *This method is deprecated in version 7.1.1. Use the addEventListener(EventListener) or removeEventListener(EventListener) methods. Plug-ins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*
void	**setMaps**(List mapList) Clears any BackingMaps that have been previously defined on this ObjectGrid and replaces them with the List of BackingMaps provided.
void	**setName**(String gridName) Sets the name of this ObjectGrid.
void	**setObjectGridAuthorization**(com.ibm.websphere.objectgrid.security.plugins.ObjectGridAuthorization ogAuthorization) Sets the ObjectGridAuthorization for this ObjectGrid instance.
void	**setPermissionCheckPeriod**(int period) Sets the permission check period.
void	**setQueryConfig**(QueryConfig queryConfig) Set the QueryConfig object for this ObjectGrid.
void	**setSecurityEnabled**() Enables the ObjectGrid security.
void	**setSubjectSource**(com.ibm.websphere.objectgrid.security.plugins.SubjectSource source) Sets the SubjectSource plugin.
void	**setSubjectValidation**(com.ibm.websphere.objectgrid.security.plugins.SubjectValidation subjectValidation) Sets the SubjectValidation for this ObjectGrid instance.
void	**setTransactionCallback**(TransactionCallback callback) Sets the TransactionCallback object.
void	**setTxIsolation**(int level) Sets the default transaction isolation level for all sessions created by the ObjectGrid.
void	**setTxTimeout**(int timeout) Sets the transaction timeout value to a specified number of seconds.

Field Detail

DEFAULT_TX_TIMEOUT_VALUE

```
static final int DEFAULT_TX_TIMEOUT_VALUE
```

The default transaction time out value of 10 minutes if no transaction time out value is set.

Since:
WXS 7.1.0.0 FIX1

See Also:
Constant Field Values

LOCAL

```
static final int LOCAL
```

Indicates the ObjectGrid is a local ObjectGrid.

See Also:
Constant Field Values

SERVER

```
static final int SERVER
```

Indicates the ObjectGrid is a server-side distributed ObjectGrid.

See Also:
Constant Field Values

CLIENT

```
static final int CLIENT
```

Indicates the ObjectGrid is a client-side distributed ObjectGrid.

See Also:
Constant Field Values

Method Detail

getSession

```
Session getSession()
                  throws ObjectGridException,
                         TransactionCallbackException
```

Gets a Session object that can be used by a single thread at a time.

It is not allowed to share this `Session` object between threads without placing a critical section around it. While the core framework allows the object to move between threads, the `TransactionCallback` and `Loader` may prevent this usage, especially in J2EE environments.

When the ObjectGrid is a local ObjectGrid, and its security is enabled, this method will use the `SubjectSource` to get a `Subject` object and then associate the `Subject` object with this session .

When the ObjectGrid is a distributed ObjectGrid (client server mode), and its security is

enabled, this method will utilize the client server security infrastructure to get a secure session.

If the `initialize()` method has not been invoked prior to the first `getSession` invocation, an implicit initialization will occur. This ensures that all of the configuration is complete before any runtime usage is required.

Returns:
An instance of `Session`
Throws:
`ObjectGridException` - if an error occurs during processing
`TransactionCallbackException` - if the `TransactionCallback` throws an exception
`IllegalStateException` - if this method is called after the `destroy()` method is called.
See Also:
`destroy()`, `initialize()`, `Session`, `SubjectSource`

---

getSession

```
Session getSession(Subject subject)
                   throws ObjectGridException,
                          TransactionCallbackException,
                          com.ibm.websphere.objectgrid.security.plugins.InvalidSubjectException
```

Allows the use of a specific `Subject` rather than use the `SubjectSource` configured on the ObjectGrid to get a Session.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided `Subject` object will not be used.

If the `initialize()` method has not been invoked prior to the first `getSession` invocation, an implicit initialization will occur. This ensures that all of the configuration is complete before any runtime usage is required.

Parameters:
`subject` - `Subject` to associate with the returned Session
Returns:
An instance of `Session`
Throws:
`ObjectGridException` - if an error occurs during processing
`TransactionCallbackException` - if the `TransactionCallback` throws an exception
`com.ibm.websphere.objectgrid.security.plugins.InvalidSubjectException` - the subject passed in is invalid based on the `SubjectValidation` mechanism.
`IllegalStateException` - if this method is called after the `destroy()` method is called.
See Also:
`destroy()`, `initialize()`, `Session`, `SubjectValidation`

---

setTransactionCallback

```
void setTransactionCallback(TransactionCallback callback)
```

Sets the TransactionCallback object.

A single cache is a single domain. All `Loader`s defined for `BackingMap`s in an ObjectGrid will normally cooperate, thus a corresponding `TransactionCallback` object needs to be set on the ObjectGrid.

A TransactionCallback that implements the `ObjectGridLifecycleListener` interface is automatically added as if the `addEventListener(EventListener)` method was called. Any previous callback which implements `ObjectGridLifecycleListener` interface is removed as if the `removeEventListener(EventListener)` method was called.

A TransactionCallback may implement the `ObjectGridPlugin` interface in order to receive enhanced ObjectGrid plug-in lifecycle method calls. The plug-in is also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:

`callback` - An instance of a TransactionCallback

Throws:

`IllegalArgumentException` - if callback is `null`

`IllegalStateException` - if this method is called after the `initialize()` method is called.

See Also:

`initialize()`, `TransactionCallback`

---

getTransactionCallback

`TransactionCallback` **`getTransactionCallback`**`()`

Retrieves the TransactionCallback object.

The TransactionCallback can be used in conjunction with the `TxID` to house transaction-specific context data, such as the connection to the database.

Returns:

the argument that was passed to the `setTransactionCallback(TransactionCallback)` method of this interface or a default `TransactionCallback` object if setTransactionCallback was not previously called for this ObjectGrid.

See Also:

`setTransactionCallback(TransactionCallback)`, `TransactionCallback`

---

setCollisionArbiter

`void` **`setCollisionArbiter`**`(com.ibm.websphere.objectgrid.revision.CollisionArbiter arbiter)`

Sets the CollisionArbiter that is responsible for arbitration of revision conflicts.

A CollistionArbiter that implements the `ObjectGridLifecycleListener` interface is automatically added as if the `addEventListener(EventListener)` method was called. Any previous arbiter which implements `ObjectGridLifecycleListener` interface is removed as if the `removeEventListener(EventListener)` method was called.

A CollisionArbiter may implement the `ObjectGridPlugin` interface in order to receive enhanced ObjectGrid plug-in lifecycle method calls. The plug-in is also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:

`arbiter` - The arbitration logic that will be used to resolve collisions.

Since:

7.1

---

getCollisionArbiter

`com.ibm.websphere.objectgrid.revision.CollisionArbiter` **`getCollisionArbiter`**`()`

Retrieves the CollisionArbiter that this grid is using to resolve revision collisions.

Returns:

The arbitration logic that is responsible for resolving revision collisions.

Since:

---

defineMap

[BackingMap](#) **defineMap**([String](#) name)

Defines a BackingMap that will be used by the application.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `initialize()` method. Also, keep in mind that the `getSession` methods implicitly call the `initialize()` method if it has yet to be called by the application.

Parameters:
`name` - the name of the map being defined.
Returns:
a BackingMap reference
Throws:
[IllegalArgumentException](#) - if name is `null`
[IllegalStateException](#) - if this method is called after the `initialize()` method is called.
See Also:
[initialize()](#), [BackingMap](#)

---

createMap

[BackingMap](#) **createMap**([String](#) name)

Creates a BackingMap, but does not associate it with this ObjectGrid.

This method is to be used in tandem with the `setMaps(List)` method, which will associate BackingMaps with this ObjectGrid. These methods are for use when configuring an ObjectGrid with the Spring Framework.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
name - the name of the map being defined.
Returns:
a BackingMap reference
Throws:
[IllegalArgumentException](#) - if name is null
[IllegalStateException](#) - if this method is called after the initialize() method is called.
Since:
WAS XD 6.0.1
See Also:
[initialize()](#), [setMaps(List)](#)

---

setMaps

void **setMaps**([List](#) mapList)

Clears any BackingMaps that have been previously defined on this ObjectGrid and replaces them with the List of BackingMaps provided.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
mapList - a list of BackingMaps to set on this ObjectGrid.
Throws:

IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.0.1

See Also:

createMap(String), initialize()

---

getListOfMapNames

List **getListOfMapNames**()

Gets the list of map names currently defined for this ObjectGrid instance.

Note, once the initialize() method is called, the List returned will not change. However, it could change if called prior to initialization. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Returns:

a List of String objects, one String per map that was previously configured by the defineMap(String) or setMaps(List) method. An empty List is returned if no maps are currently defined.

See Also:

defineMap(String), initialize(), setMaps(List)

---

getMap

BackingMap **getMap**(String name)

Returns a BackingMap previously configured by calling the defineMap(String) or setMaps(List) method.

Parameters:

name - the same name that was used as an argument to the defineMap(String) or createMap(String) method. A null reference is returned if a map is not associated with this ObjectGrid for the specified map name.

Returns:

the BackingMap instance

See Also:

createMap(String), defineMap(String), setMaps(List)

---

initialize

void **initialize**()
throws ObjectGridException

Begins the bootstrapping of the ObjectGrid and Session instances.

After this method has been invoked, the configuration of the ObjectGrid is considered complete and is ready for runtime usage. Any additional configuration method invocations, such as defineMap(String), will result in an exception. This method is considered optional since the first call to one of the getSession methods will perform an implicit initialization.

Throws:

ObjectGridException - if an error occurs during processing

---

addEventListener

void **addEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener)

Adds an EventListener.

Significant events will be communicated to interested listeners through the ObjectGridEventListener and ObjectGridLifecycleListener callback interface. Multiple event listeners are allowed to be registered, with no implied ordering of event notifications.

Note, this method is allowed to be invoked before and after the initialize() method.

Object grid plug-ins (TransactionCallback, CollisionArbiter) that implement the ObjectGridLifecycleListener are automatically added as lifecycle listeners when added to the ObjectGrid.

Parameters:
listener - An instance of ObjectGridEventListener or ObjectGridLifecycleListener
Throws:
IllegalArgumentException - if listener is null or not an instance of ObjectGridEventListener, ObjectGridLifecycleListener.
IllegalStateException - if this method is called during initialization by one of the configured plugins and the ObjectGrid runtime is not in a usable state to initialize the ObjectGridEventListener.
See Also:
ObjectGridEventListener, ObjectGridLifecycleListener, EventListener

---

addEventListener

void **addEventListener**(ObjectGridEventListener listener)

Deprecated. *This method is deprecated in version 7.1.1, use the addEventListener(EventListener) method.*

Provided for compatibility with old releases, use the addEventListener(EventListener) method.

Parameters:
listener -

---

removeEventListener

void **removeEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener)

Removes an EventListener.

This method removes an ObjectGridEventListener or ObjectGridLifecycleListener that was previously added to this object using the addEventListener(ObjectGridEventListener) or setEventListeners(List) method. If the desired ObjectGridEventListener is not found, no error will be returned.

Note, this method is allowed to be invoked before and after the initialize() method. Object grid plug-ins (TransactionCallback, CollisionArbiter) that implement the ObjectGridLifecycleListener are automatically removed as lifecycle listeners when removed from the ObjectGrid.

Parameters:
listener - An instance of ObjectGridEventListener or ObjectGridLifecycleListener
Throws:
IllegalArgumentException - if listener is null or not an instance of ObjectGridEventListener, ObjectGridLifecycleListener
See Also:
addEventListener(EventListener), ObjectGridEventListener, EventListener

---

removeEventListener

void **removeEventListener**(ObjectGridEventListener listener)

Deprecated. *This method is deprecated in version 7.1.1, use the removeEventListener(EventListener) method.*

Provided for compatibility with old releases, use the removeEventListener(EventListener) method.

Parameters:
listener -

---

setEventListeners

@Deprecated
void **setEventListeners**(List listeners)

Deprecated. *This method is deprecated in version 7.1.1. Use the addEventListener(EventListener) or removeEventListener(EventListener) methods. Plug-ins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*

This overwrites the current list of EventListeners and replaces it with the supplied List of EventListeners

Note, this method is allowed to be invoked before and after the initialize() method.

Parameters:
listeners - List of ObjectGridEventListeners and ObjectGridLifecycleListener instances
Throws:
ClassCastException - if one of the elements in the provided list is not an instance of ObjectGridEventListener
IllegalArgumentException - if listeners is null, contains a null reference, or contains an instance of a type other than ObjectGridEventListener and ObjectGridLifecycleListener
IllegalStateException - if this method is called during initialization by one of the configured plugins and the ObjectGrid runtime is not in a usable state to initialize the ObjectGridEventListener objects.
See Also:
EventListener, addEventListener(EventListener), removeEventListener(EventListener)

---

getEventListeners

List **getEventListeners**()

Returns the current list of EventListeners.

Returns:
The current list of EventListeners.
See Also:
addEventListener(EventListener), EventListener, ObjectGridEventListener, ObjectGridLifecycleListener

---

getName

String **getName**()

Gets the name of this ObjectGrid.

This method is useful for authorization as all Maps are prefixed with the ObjectGrid name.

Returns:
The name of the ObjectGrid.
See Also:
setName(String)

---

setName

void **setName**(String gridName)

Sets the name of this ObjectGrid. Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
gridName - The ObjectGrid name to use.
Throws:
IllegalArgumentException - if gridName is null
IllegalStateException - if this method is called after the initialize() method is called.

---

reserveSlot

int **reserveSlot**(String containerName)

Allows plugins on this ObjectGrid to reserve slots for use to store context data.

Currently the TxID object is the only object that uses slots for storing context data. TxID slots are used for storing transactional context data.

Once a slot is reserved, the slot assignment is permanent and cannot be given back. Note, this method is allowed to be invoked before and after the initialize() method.

Parameters:
containerName - The name of the Object with the slots. Currently TxID.SLOT_NAME is the only supported value for this argument.
Returns:
The slot index to use.
Throws:
IllegalArgumentException - if containerName is not TxID.SLOT_NAME.
See Also:
TxID.SLOT_NAME, TxID.getSlot(int), TxID.putSlot(int, Object)

---

setSubjectValidation

void **setSubjectValidation**(com.ibm.websphere.objectgrid.security.plugins.SubjectValidation subjectValidation)

Sets the SubjectValidation for this ObjectGrid instance.

Passing null to this method removes a previously set SubjectValidation object from an earlier invocation of this method and indicates that this ObjectGrid is not associated with a SubjectValidation object.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided SubjectValidation object will not be used.

A SubjectValidation plugin can be used to validate the Subject object passed in is a valid Subject. Please refer to SubjectValidation for more details.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
subjectValidation - the SubjectValidation plugin
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
getSession(Subject), initialize(), SubjectValidation

---

setAuthorizationMechanism

void **setAuthorizationMechanism**(int authMechanism)

Sets the authorization mechanism.

If this method is not invoked, the default authorization mechanism is SecurityConstants.AUTHORIZATION_MECHANISM_JAAS.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provide authorization mechanism will not be used.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
authMechanism - the authorization mechanism, must be one of the final static variable on the SecurityConstants class.
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize(), SecurityConstants.AUTHORIZATION_MECHANISM_CUSTOM, SecurityConstants.AUTHORIZATION_MECHANISM_JAAS

setSecurityEnabled

void **setSecurityEnabled**()

Enables the ObjectGrid security.

Security on the ObjectGrid level refers to ObjectGrid authorizations.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize()

---

isSecurityEnabled

boolean **isSecurityEnabled**()

Checks whether security is enabled on this ObjectGrid or not.

Security on the ObjectGrid level refers to ObjectGrid authorizations. Security is disabled by default.

Returns:
true if security is enabled on this ObjectGrid; false otherwise.
See Also:
setSecurityEnabled()

---

setPermissionCheckPeriod

void **setPermissionCheckPeriod**(int period)

Sets the permission check period.

This method takes a single parameter indicating how often the customer wants to check the permission used to allow a client access. If the parameter is 0 then every single authorized operation call will ask the authorization mechanism, either JAAS authorization or custom authorization to check if the current Subject has permission. This approach may be prohibitively expensive from a performance point of view depending on the authorization implementation, but if it is required then you can do it. Alternatively, if the parameter is > 0 then it indicates the number of seconds to cache a set of permissions before returning to the authorization mechanism to refresh them. This mechanism provides much better performance, but you run the risk that if the back-end permissions are changed during this time, the ObjectGrid will possibly allow or prevent access even though the back-end security provider has been modified.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
period - the permission check period in seconds.
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize()

---

setSubjectSource

void **setSubjectSource**(com.ibm.websphere.objectgrid.security.plugins.SubjectSource source)

Sets the SubjectSource plugin.

Passing null to this method removes a previously set SubjectSource object from an earlier invocation of this method and indicates that this ObjectGrid is not associated with a SubjectSource object.

A SubjectSource plugin can be used to get a Subject object from the environment to represent the ObjectGrid client.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided SubjectSource object will not be used.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
source - the SubjectSource plugin
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize(), SubjectSource

---

setTxTimeout

void **setTxTimeout**(int timeout)

Sets the transaction timeout value to a specified number of seconds.

Any transaction that is started by use of a Session returned by one of the getSession methods on this interface must complete within the number of seconds specified by the transaction timeout parameter of this method. The timeout value is the maximum number of seconds the transaction is allowed to execute. If a transaction executes longer than this amount, a TransactionTimeoutException is thrown and the transaction is rolled back even if commit is requested.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

The transaction timeout is used by any transaction started by a Session that is returned by the getSession methods of this interface. Since this method must be called prior to getSession method to avoid IllegalStateException, this method only affects transactions that are started after this method is called. If this method is never called, the transaction is allowed unlimited amount of time to complete.

Parameters:
timeout - is the transaction timeout value in seconds. Use a value of 0 to indicate a transaction is allowed unlimited amount of time so that no TransactionTimeoutException ever occurs.
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
Since:
WAS XD 6.0.1
See Also:
initialize(), Session.TRANSACTION_NO_TIMEOUT, Session.setTransactionTimeout(int), TransactionTimeoutException

---

getTxTimeout

int **getTxTimeout**()

Gets transaction timeout setting for this ObjectGrid instance.

Returns:
timeout value that was passed to the setTxTimeout(int) method or 0 if setTxTimeout was never called.
Since:
WAS XD 6.0.1
See Also:

setTxTimeout(int)

---

setTxIsolation

void **setTxIsolation**(int level)

Sets the default transaction isolation level for all sessions created by the ObjectGrid. The constants defined in the Session interface are the possible transaction isolation levels. The default is Session.TRANSACTION_REPEATABLE_READ.

Parameters:
level - one of the following Session constants: Session.TRANSACTION_READ_UNCOMMITTED, Session.TRANSACTION_READ_COMMITTED or Session.TRANSACTION_REPEATABLE_READ
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
IllegalArgumentException - if this method is includes and invalid transaction isolation level.
Since:
7.1.1

---

getTxIsolation

int **getTxIsolation**()

Retrieves the default transaction isolation level.

Returns:
the current transaction isolation level.
Since:
7.1.1
See Also:
setTxIsolation(int)

---

destroy

void **destroy**()

Destroys this instance.

This method should be invoked when the ObjectGrid is no longer being used. When this method is called, the ObjectGrid can free up any resources it is using. No new Sessions can be created or used after the destroy() has been invoked. Any in-flight Sessions will be allowed to continue, if the resources are still available to complete processing.

---

getSession

Session **getSession**(CredentialGenerator credGen)
throws ObjectGridException,
TransactionCallbackException

Get a session using a CredentialGenerator.

This method can only be called by the ObjectGrid client in an ObjectGrid client server environment. If ObjectGrid is used in a local model, that is, within the same JVM with no client or server existing, getSession(Subject) or the SubjectSource plugin should be used to secure the ObjectGrid.

If the initialize() method has not been invoked prior to the first getSession invocation, an implicit initialization will occur. This ensures that all of the configuration is complete before any runtime usage is required.

Parameters:
credGen - A CredentialGenerator for generating a credential for the session returned.
Returns:
An instance of Session
Throws:
ObjectGridException - if an error occurs during processing

TransactionCallbackException - if the TransactionCallback throws an exception
IllegalStateException - if this method is called after the destroy() method is called.

Since:

WAS XD 6.0.1

See Also:

destroy(), initialize(), CredentialGenerator, Session

---

setQueryConfig

void **setQueryConfig**(QueryConfig queryConfig)

Set the QueryConfig object for this ObjectGrid. A QueryConfig object provides query configurations for executing object queries over the maps in this ObjectGrid.

Parameters:

queryConfig - The QueryConfig to associate with this ObjectGrid instance.

Throws:

IllegalArgumentException - if queryConfig is null.
IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.1

See Also:

QueryConfig

---

registerEntities

void **registerEntities**(URL entityXML)

Registers one ore more entities from an entity XML file.

Entity registration is required prior to ObjectGrid initialization to bind an Entity with a BackingMap and any defined indices.

This method may be called multiple times.

Parameters:

entityXML - the URL of the entity XML that defines the entities.

Throws:

IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.1

---

registerEntities

void **registerEntities**(Class[] entities)

Register one or more entities based on the class metadata.

Entity registration is required prior to ObjectGrid initialization to bind an Entity with a BackingMap and any defined indices.

This method may be called multiple times.

Parameters:

entities - one or more annotated entity classes to register as entities.

Throws:

IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.1

---

getObjectGridType

int **getObjectGridType**()

Returns the type of ObjectGrid.

The return value is equivalent to one of the constants declared on this interface, LOCAL, SERVER, or CLIENT.

Returns:
the ObjectGrid type
Since:
WAS XD 6.1

---

setObjectGridAuthorization

void **setObjectGridAuthorization**(com.ibm.websphere.objectgrid.security.plugins.ObjectGridAuthorization ogAuthorization)

Sets the ObjectGridAuthorization for this ObjectGrid instance.

Passing null to this method removes a previously set ObjectGridAuthorization object from an earlier invocation of this method and indicates that this ObjectGrid is not associated with a ObjectGridAuthorization object.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided ObjectGridAuthorization object will not be used.

A ObjectGridAuthorization plugin can be used to authorize access to the ObjectGrid and maps. Please refer to ObjectGridAuthorization for more details.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
ogAuthorization - the ObjectGridAuthorization plugin
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
Since:
WAS XD 6.1
See Also:
initialize(), ObjectGridAuthorization

---

setAccessByCreatorOnlyMode

void **setAccessByCreatorOnlyMode**(int accessByCreatorOnlyMode)

Set the "access by creator only" mode.

Enabling "access by creator only" mode ensures that only the user (represented by the Principals associated with it), who inserts the record into the map, can access (read, update, invalidate, and remove) the record.

The "access by creator only" mode can be disabled, or can complement the ObjectGrid authorization model, or it can supersede the ObjectGrid authorization model. The default value is disabled: SecurityConstants.ACCESS_BY_CREATOR_ONLY_DISABLED.

Parameters:
accessByCreatorOnlyMode - the access by creator mode.
Since:
WAS XD 6.1 FIX3
See Also:
SecurityConstants.ACCESS_BY_CREATOR_ONLY_DISABLED,
SecurityConstants.ACCESS_BY_CREATOR_ONLY_COMPLEMENT,
SecurityConstants.ACCESS_BY_CREATOR_ONLY_SUPERSEDE

---

getState

com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener.State **getState**()

Retrieve the current life cycle state of this ObjectGrid.

Returns:

the current state.

Since:

7.1.1

---

com.ibm.websphere.objectgrid

Class NoActiveTransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.NoActiveTransactionException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class NoActiveTransactionException
extends ObjectGridException
```

An exception indicating there is no active transaction.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

NoActiveTransactionException() Constructs a new NoActiveTransactionException with `null` as its detail message.
NoActiveTransactionException(String message) Constructs a new NoActiveTransactionException with the specified detail message.
NoActiveTransactionException(String message, Throwable cause) Constructs a new NoActiveTransactionException with the specified detail message and cause.
NoActiveTransactionException(Throwable cause) Constructs a new NoActiveTransactionException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`()`

Constructs a new NoActiveTransactionException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

`ObjectGridException.initCause(Throwable)`

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`(String message)`

Constructs a new NoActiveTransactionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`(String message,`
`Throwable cause)`

Constructs a new NoActiveTransactionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this NoActiveTransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`(Throwable cause)`

Constructs a new NoActiveTransactionException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for NoActiveTransactionExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class LockTimeoutException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException
          com.ibm.websphere.objectgrid.LockTimeoutException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

LockDeadlockException, LockInternalFailureException

```
public class LockTimeoutException
extends LockException
```

This exception is used by the lock manager to indicate that the maximum wait time for a lock has been exceeded. The timeout may or may not be the result of a deadlock. If it is a deadlock, the timeout is used to break the deadlock.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

LockTimeoutException() Constructs a new LockTimeoutException with `null` as its detail message.
LockTimeoutException(String message) Constructs a new LockTimeoutException with the specified detail message.
LockTimeoutException(String message, Throwable cause) Constructs a new LockTimeoutException with the specified detail message and cause.
LockTimeoutException(Throwable cause) Constructs a new LockTimeoutException with a specified cause.

Method Summary

void	**forceJavaCore**()
St	**getLockRequestQueueDetails**()

[ring](#)	Provides detailed information about the state of the lock queue when the lock timeout occurred.
[String](#)	**getMessage**() Returns the detail message string of this exception.
void	**setLockRequestQueueDetails**([String](#) string) Sets the details of the lock requests on the lock request queue at the time the lock timeout occurred.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockTimeoutException

public **LockTimeoutException**()

Constructs a new LockTimeoutException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

LockTimeoutException

public **LockTimeoutException**(String message)

Constructs a new LockTimeoutException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
ObjectGridException.initCause(Throwable), getMessage()

LockTimeoutException

public **LockTimeoutException**(String message,
Throwable cause)

Constructs a new LockTimeoutException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LockTimeoutException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `getMessage()`

LockTimeoutException

`public` **`LockTimeoutException`**`(`Throwable` `cause)`

Constructs a new LockTimeoutException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LockTimeoutExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

Method Detail

getLockRequestQueueDetails

`public` `String` **`getLockRequestQueueDetails`**`()`

Provides detailed information about the state of the lock queue when the lock timeout occurred.

Returns:

the argument that was passed to the `setLockRequestQueueDetails(String)` method of this class or `null` if the `setLockRequestQueueDetails` method was not previously called for this object.

---

forceJavaCore

`public void` **`forceJavaCore`**`()`

---

setLockRequestQueueDetails

`public void` **`setLockRequestQueueDetails`**`(`String` `string)`

Sets the details of the lock requests on the lock request queue at the time the lock timeout occurred.

Parameters:

`string` - the details of lock requests on the lock request queue at the time the lock timeout occurred.

---

getMessage

public [String](#) **getMessage**()

Returns the detail message string of this exception. The returned String includes the request queue details as well as the message provided to the constructor.

Overrides:

[getMessage](#) in class [Throwable](#)

Returns:

the detail message string of this object instance

---

com.ibm.websphere.objectgrid

Class LockStrategy

java.lang.Object

  com.ibm.websphere.objectgrid.LockStrategy

```
public final class LockStrategy
extends Object
```

LockStrategy provides an enumerated type idiom for use on the `BackingMap.setLockStrategy(LockStrategy)` method. It determines whether or not a lock manager is needed for a BackingMap and if so, whether to use a optimistic or pessimistic locking strategy.

Since:

WAS XD 6.0, XC10

See Also:

`BackingMap.setLockStrategy(LockStrategy)`

Field Summary

static LockStrategy	**NONE** NONE indicates internal LockManager use is not needed since concurrency control is provided outside of ObjectGrid either by a persistence manager using objectgrid as a side cache, the application, or by a Loader plugin (for example, uses database locks to control concurrency).
static LockStrategy	**OPTIMISTIC** OPTIMISTIC is typically used for a map that does not have a Loader plugin, the map is read mostly, and locking is neither provided by a persistence manager using ObjectGrid as a side cache or by the application itself.
s	

static LockStrategy	**PESSIMISTIC** PESSIMISTIC is typically used for a map that does not have a Loader plugin and locking is neither provided by a persistence manager using ObjectGrid as a side cache, by a Loader plugin, or by the application itself.

Method Summary

String	**toString**() Returns a string representation of the `LockStrategy`.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

NONE

```
public static final LockStrategy NONE
```

NONE indicates internal LockManager use is not needed since concurrency control is provided outside of ObjectGrid either by a persistence manager using objectgrid as a side cache, the application, or by a Loader plugin (for example, uses database locks to control concurrency).

OPTIMISTIC

```
public static final LockStrategy OPTIMISTIC
```

OPTIMISTIC is typically used for a map that does not have a Loader plugin, the map is read mostly, and locking is neither provided by a persistence manager using ObjectGrid as a side cache or by the application itself. For this strategy, an exclusive lock is obtained on a map entry being inserted, updated, or removed at commit time. The lock ensures version information cannot be changed by another transaction while the transaction being committed is performing an optimistic version check.

PESSIMISTIC

```
public static final LockStrategy PESSIMISTIC
```

PESSIMISTIC is typically used for a map that does not have a Loader plugin and locking is neither provided by a persistence manager using ObjectGrid as a side cache, by a Loader plugin, or by the application itself. It is typically used when optimistic approach fails too often since there are update transactions that frequently collide on the same

map entry (e.g. not a read mostly map or large number of clients accessing a small map).

Method Detail

toString

public [String](#) **toString**()

Returns a string representation of the `LockStrategy`.

Overrides:

[toString](#) in class [Object](#)

Returns:

a string representation of the `LockStrategy`.

---

com.ibm.websphere.objectgrid

Class LockInternalFailureException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException
          com.ibm.websphere.objectgrid.LockTimeoutException
            com.ibm.websphere.objectgrid.LockInternalFailureException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **LockInternalFailureException**
extends LockTimeoutException

This exception is used by the lock manager to indicate it detected some internal programming error while processing a lock or unlock request.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

LockInternalFailureException() Constructs a new LockInternalFailureException with `null` as its detail message.
LockInternalFailureException(String message) Constructs a new LockInternalFailureException with the specified detail message.
LockInternalFailureException(String message, Throwable cause) Constructs a new LockInternalFailureException with the specified detail message and cause.
LockInternalFailureException(Throwable cause) Constructs a new LockInternalFailureException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.LockTimeoutException
forceJavaCore, getLockRequestQueueDetails, getMessage, setLockRequestQueueDetails

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockInternalFailureException

public **LockInternalFailureException**()

Constructs a new LockInternalFailureException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

LockInternalFailureException

public **LockInternalFailureException**(String message)

Constructs a new LockInternalFailureException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
ObjectGridException.initCause(Throwable), LockTimeoutException.getMessage()

LockInternalFailureException

public **LockInternalFailureException**(String message,
Throwable cause)

Constructs a new LockInternalFailureException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LockInternalFailureException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
ObjectGridException.getCause(), LockTimeoutException.getMessage()

LockInternalFailureException

`public` **`LockInternalFailureException`**`(`[`Throwable`]` cause)`

Constructs a new LockInternalFailureException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LockInternalFailureExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class LockException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

LockTimeoutException

```
public class LockException
extends ObjectGridException
```

A general locking exception indicating something went wrong with locking operations.

Since:

WAS XD 6.0, XC10

See Also:

LockTimeoutException, Serialized Form

Constructor Summary

LockException() Constructs a new LockException with `null` as its detail message.
LockException(String message) Constructs a new LockException with the specified detail message.
LockException(String message, Throwable cause) Constructs a new LockException with the specified detail message and cause.
LockException(Throwable cause) Constructs a new LockException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockException

```
public LockException()
```

Constructs a new LockException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

LockException

```
public LockException(String message)
```

Constructs a new LockException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

LockException

```
public LockException(String message,
                     Throwable cause)
```

Constructs a new LockException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LockException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

LockException

```
public LockException(Throwable cause)
```

Constructs a new LockException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LockExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for

later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class LockDeadlockException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException
          com.ibm.websphere.objectgrid.LockTimeoutException
            com.ibm.websphere.objectgrid.LockDeadlockException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **LockDeadlockException**
extends LockTimeoutException

This exception is used by the lock manager to indicate that it detected a deadlock. It prevents the deadlock by throwing this exception. Typically, this deadlock is a result of the following scenario: one transaction owns a weaker lock as a result of getting a map entry, and then, at commit time, the transaction attempts to promote the weaker lock to a stronger lock in order to apply the changes to the data store. For example, two transactions try to promote from shared locks to exclusive locks but each transaction already owns a shared lock.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

LockDeadlockException() Constructs a new LockDeadlockException with `null` as its detail message.
LockDeadlockException(String message) Constructs a new LockDeadlockException with the specified detail message.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.LockTimeoutException
forceJavaCore, getLockRequestQueueDetails, getMessage, setLockRequestQueueDetails

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockDeadlockException

public **LockDeadlockException**()

Constructs a new LockDeadlockException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

LockDeadlockException

public **LockDeadlockException**(String message)

Constructs a new LockDeadlockException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), LockTimeoutException.getMessage()

---

com.ibm.websphere.objectgrid

Class KeyNotFoundException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.KeyNotFoundException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **KeyNotFoundException**
extends ObjectGridException

Normally, record not found means a null is returned. However, sometimes on the explicit operation methods like update methods, we can figure that the record isn't there and then we throw this exception.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

KeyNotFoundException() Constructs a new KeyNotFoundException with null as its detail message.
KeyNotFoundException(String message) Constructs a new KeyNotFoundException with the specified detail message.
KeyNotFoundException(String message, Throwable cause) Constructs a new KeyNotFoundException with the specified detail message and cause.
KeyNotFoundException(Throwable cause) Constructs a new KeyNotFoundException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

KeyNotFoundException

public **KeyNotFoundException**()

Constructs a new KeyNotFoundException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

KeyNotFoundException

public **KeyNotFoundException**(String message)

Constructs a new KeyNotFoundException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

KeyNotFoundException

public **KeyNotFoundException**(String message,
Throwable cause)

Constructs a new KeyNotFoundException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this KeyNotFoundException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

KeyNotFoundException

public **KeyNotFoundException**(Throwable cause)

Constructs a new KeyNotFoundException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for KeyNotFoundExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that

the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Interface JavaMap

All Superinterfaces:

Map

```
public interface JavaMap
extends Map
```

This interface is a handle to a named Map. Maps should have homogeneous keys and values. An instance of this JavaMap can only be used by the thread that is currently associated with the Session that was used to get this JavaMap instance. Both Session and JavaMap objects are not allowed to be shared by multiple threads concurrently.

Users can get an instance of `JavaMap` from an instance of `ObjectMap` by calling `ObjectMap.getJavaMap()`. There are two main differences between `JavaMap` and `ObjectMap`:

- JavaMap extends java.util.Map. Therefore, users can cast an instance of `JavaMap` to java.util.Map if they want.
- The methods in `JavaMap` are defined to throw Exceptions similar to those defined on the `java.util.Map` interface that is `ObjectGridRuntimeException`, which is a subclass of `java.lang.RuntimeException` is used for error conditions. The methods in `ObjectMap` are defined to throw `ObjectGridException`s, which are checked exceptions.

The only methods that are supported from the `java.util.Map` interface are:

- containsKey(Object)
- get(Object)
- put(Object, Object)
- putAll(Map)
- remove(Object)
- clear()

All other methods on the `java.util.Map` interface will throw `java.lang.UnsupportedOperationException`.

Since:
WAS XD 6.0, XC10

See Also:
ObjectMap, ObjectMap.getJavaMap(), Map, BackingMap.setCopyMode(CopyMode, Class), BackingMap.setLockStrategy(LockStrategy)

Nested Class Summary

Nested classes/interfaces inherited from interface java.util.Map
Map.Entry<K,V>

Method Summary

void	**clear**() Clear all keys from the Map.
void	**clearCopyMode**() Resets the CopyMode back to the one in the BackingMap.
boolean	**containsKey**(Object key) Returns `true` if this map contains a mapping for the specified key.
boolean	**containsValue**(Object value) This method of the `java.util.Map` interface is not supported.
Set	**entrySet**() This method of the `java.util.Map` interface is not supported.
void	**flush**() Pushes the current set of changes for the `JavaMap` instance to the `Loader` without committing the changes.
Object	**get**(Object key) Retrieves the object from the cache at the given key.
List	**getAll**(List keyList) Gets a list of entries from the map.
List	**getAllForUpdate**(List keyList) Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries.
com.ibm.websphere.pro	

JectorMetadata	**getEntityMetadata**() Retrieve the metadata for the entity associated with this map.
Object	**getForUpdate**(Object key) Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry.
Object	**getIndex**(String name) Returns a reference to the named index that can be used with this Map.
String	**getName**() Returns the name of the JavaMap as defined by the configuration.
Object	**getNextKey**(long timeout) Retrieves a key off the map in first-in-first-out (FIFO) insert order.
void	**insert**(Object key, Object value) Performs an explicit insert of a given entry.
void	**invalidate**(Object key, boolean isGlobal) Invalidates an entry in the cache based on the key parameter.
void	**invalidateAll**(Collection keyList, boolean isGlobal) Invalidate a set of cache entries based on the Collection of keys provided.
boole	**isEmpty**() This method of the `java.util.Map` interface is not supported.

an	
Set	**keySet**() This method of the `java.util.Map` interface is not supported.
Object	**put**(Object key, Object value) Puts the Object value into the cache at location represented by key.
void	**putAll**(Map map) Puts each of the Object values into the cache at location represented by the corresponding key contained in the Map.
Object	**remove**(Object key) Removes the Object value from the cache represented by key.
void	**removeAll**(Collection keyList) Batch remove from the Map.
void	**setCopyMode**(CopyMode copyMode, Class valueInterface) Allows the CopyMode for the Map to be overridden on this map on this session only.
int	**setTimeToLive**(int ttl) Establishes the number of seconds that any given cache entry can live for, which is refered to as "time to live" or TTL.
int	**size**() This method of the `java.util.Map` interface is not supported.
void	**touch**(Object key) Updates the last access time in the BackingMap without retrieving the value to the JavaMap.
void	**update**(Object key, Object value) Performs an explicit update of a given entry.
Collection	**values**() This method of the `java.util.Map` interface is not supported.

Methods inherited from interface java.util.Map
equals, hashCode

Method Detail

getName

`String` **`getName`**`()`

Returns the name of the JavaMap as defined by the configuration.

Returns:
name of JavaMap

---

getForUpdate

`Object` **`getForUpdate`**`(Object key)`
`throws ObjectGridRuntimeException`

Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry. See `LockStrategy.PESSIMISTIC` for additional information. Whether or not a copy of the object is returned is determined by the `CopyMode` setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if the value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getForUpdate(Object)` for additional specification details.

Parameters:
`key` - The entry to fetch
Returns:
the value retrieved for update or `null`
Throws:
`IllegalArgumentException` - if key is `null`
`ObjectGridRuntimeException` - if an error occurs during processing
See Also:
`containsKey(Object)`, `get(Object)`, `CopyMode`, `LockStrategy.PESSIMISTIC`, `ObjectMap.getForUpdate(Object)`

---

getAll

`List` **`getAll`**`(List keyList)`
`throws ObjectGridRuntimeException`

Gets a list of entries from the map.

If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getAll(List)` for additional specification details.

Parameters:
`keyList` - A list of keys for identifying which entries to fetch
Returns:
a list of values
Throws:
`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing
See Also:
`get(Object)`, `ObjectMap.getAll(List)`

---

getAllForUpdate

```
List getAllForUpdate(List keyList)
                     throws ObjectGridRuntimeException
```

Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries. See `LockStrategy.PESSIMISTIC` for additional information. If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getAllForUpdate(List)` for additional specification details.

Parameters:
`keyList` - A list of keys for identifying which entries to fetch
Returns:
a list of values
Throws:
`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing
See Also:
`getAll(List)`, `getForUpdate(Object)`, `LockStrategy.PESSIMISTIC`, `ObjectMap.getAllForUpdate(List)`

---

removeAll

```
void removeAll(Collection keyList)
               throws ObjectGridRuntimeException
```

Batch remove from the Map. If a key in the list cannot be found, it will be ignored.

See `ObjectMap.removeAll(Collection)` for additional specification details.

Parameters:
`keyList` - A list of keys for identifying which entries to remove
Throws:
`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

remove(Object), ObjectMap.removeAll(Collection)

---

invalidate

```
void invalidate(Object key,
                boolean isGlobal)
                throws ObjectGridRuntimeException
```

Invalidates an entry in the cache based on the key parameter.

If the key's value has changes pending in the JavaMap, it is the application's responsibility to flush these changes to the Loader before invalidation. If a flush is not performed prior to invoking the invalidate operation, all pending changes for this key will be removed from the JavaMap. If the key cannot be found in the map, it will be ignored.

The isGlobal parameter is used to indicate which cache level is used to invalidate the entries. If isGlobal is true, when the transaction is committed, the key is removed from the BackingMap also. If a subsequent get operation is performed, the BackingMap will be skipped and the Loader will be used to get the data. If isGlobal is false, the entry is only invalidated in the JavaMap (transactional cache). If a subsequent get operation is performed, the BackingMap can be used; and, if it's not in the BackingMap, the Loader will be used to get the data.

A typical use of isGlobal being false is when a large number of records are touched in a transaction and the application wants to evict records that are no longer used in the cache.

See `ObjectMap.invalidate(Object, boolean)` for additional specification details.

Parameters:

`key` - Object representing the key to be used for cache entry invalidation
`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the JavaMap (false).

Throws:

`IllegalArgumentException` - if key is `null`
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`ObjectMap.invalidate(Object, boolean)`

---

invalidateAll

```
void invalidateAll(Collection keyList,
                   boolean isGlobal)
                   throws ObjectGridRuntimeException
```

Invalidate a set of cache entries based on the Collection of keys provided. If a key in the collection cannot be found, it will be ignored.

See `ObjectMap.invalidateAll(Collection, boolean)` for additional specification details.

Parameters:

`keyList` - A Collection of keys representing the entries to be invalidated
`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the JavaMap (false).

Throws:

`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`invalidate(Object, boolean)`, `ObjectMap.invalidateAll(Collection, boolean)`

---

setTimeToLive

```
int setTimeToLive(int ttl)
```

Establishes the number of seconds that any given cache entry can live for, which is refered to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this `JavaMap`, any previous value set by the `BackingMap.setTimeToLive(int)` method is overridden for this JavaMap. If this method is never called on the JavaMap, the TTL value from the BackingMap setting is used by default. If TTL is never set on the BackingMap, the cache entry can live "forever".

This method can only be used when the `TTLType` is set to `LAST_ACCESS_TIME` on the BackingMap. If this method is called on the JavaMap and the `TTLType` is something other than `LAST_ACCESS_TIME`, an IllegalStateException is thrown.

Parameters:

`ttl` - is the time-to-live value in seconds. The value must be >= 0. A value of 0 is used to indicate the cache entry can live "forever". Use of the constant `ObjectMap.TTL_FOREVER` is recommended when "forever" is desired.

Returns:

previous time-to-live value in seconds. The constant `ObjectMap.TTL_FOREVER` can be used to determine if the previous TTL was set to "forever".

Throws:

`IllegalArgumentException` - if seconds argument is < 0.

`IllegalStateException` - if `BackingMap.getTtlEvictorType()` returns anything other than `TTLType.LAST_ACCESS_TIME`.

See Also:

`BackingMap.setTimeToLive(int)`, `ObjectMap.setTimeToLive(int)`, `TTLType.LAST_ACCESS_TIME`

---

update

```
void update(Object key,
            Object value)
            throws ObjectGridRuntimeException
```

Performs an explicit update of a given entry.

A `get` operation is not required prior to invoking the `update` method (unlike the `put` method). Also, an `update` invocation will never insert a new record. If a the map's `LockStrategy` is `LockStrategy.OPTIMISTIC` this method will implicitly get the entry so as to have the version value of the object for when this method was invoked. Whether or not a copy of the object is made when transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If a key cannot be found in the map during commit, a TransactionException will be thrown.

See `ObjectMap.update(Object, Object)` for additional specification details.

Parameters:

`key` - Identifies the entry to be updated

`value` - The updated value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.

`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`insert(Object, Object)`, `put(Object, Object)`, `CopyMode`, `LockStrategy.OPTIMISTIC`,

ObjectMap.update(Object, Object)

---

insert

```
void insert(Object key,
            Object value)
            throws ObjectGridRuntimeException
```

Performs an explicit insert of a given entry.

The key must not exist before executing this method. Also, an `insert` invocation will never update an existing record. Whether or not a copy of the object is made when a transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If the key is already in the map, a TransactionException will be thrown during commit.

See `ObjectMap.insert(Object, Object)` for additional specification details.

Parameters:

`key` - Identifies the entry to be inserted
`value` - The value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`put(Object, Object)`, `update(Object, Object)`, `CopyMode`, `ObjectMap.insert(Object, Object)`

---

getIndex

```
Object getIndex(String name)
                throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                       com.ibm.websphere.objectgrid.IndexNotReadyException,
                       UnsupportedOperationException
```

Returns a reference to the named index that can be used with this Map. This index cannot be shared between threads and works on the same rules as `Session`. The returned value should be cast to the right index interface such as `MapIndex`, `MapRangeIndex` or a custom index interface such as a geo spatial index.

Parameters:

`name` - The index name

Returns:

A reference to the index, it must be cast to the appropriate index interface.

Throws:

`IndexUndefinedException` - if the index is not defined on the BackingMap
`IndexNotReadyException` - if the index is a dynamic index and it is not ready
`UnsupportedOperationException` - if the map is a distributed map

Since:

WAS XD 6.0.1

---

flush

```
void flush()
          throws ObjectGridRuntimeException
```

Pushes the current set of changes for the `JavaMap` instance to the `Loader` without committing the changes. The changes are not propagated to the `BackingMap` either. This is useful for re-priming the `Loader`'s data without committing the current transaction and

starting over.

Throws:

ObjectGridRuntimeException - if an error occurs during processing

See Also:

Session.flush(), ObjectMap.flush()

---

size

`int` **`size`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:

size in interface Map

Returns:

the number of key-value mappings in this map.

Throws:

UnsupportedOperationException - indicating this method is not supported

---

isEmpty

`boolean` **`isEmpty`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:

isEmpty in interface Map

Returns:

`true` if this map contains no key-value mappings.

Throws:

UnsupportedOperationException - indicating this method is not supported

---

containsKey

`boolean` **`containsKey`**`(Object key)`

Returns `true` if this map contains a mapping for the specified key. ObjectGrid does not support null keys. If you configured the map to support `null` values, this method can be used to determine whether a key is contained in the map or not.

This API does not hold any locks when using pessimistic locking.

See ObjectMap.containsKey(Object) for additional specification details.

Specified by:

containsKey in interface Map

Parameters:

`key` - key whose presence in this map is to be tested.

Returns:

`true` if this map contains a mapping for the specified key.

Throws:

IllegalArgumentException - if `null` key parameter is passed in

ObjectGridRuntimeException - if an error occurs during processing

See Also:

ObjectMap.containsKey(Object)

---

containsValue

`boolean` **`containsValue`**`(Object value)`

This method of the `java.util.Map` interface is not supported.

Specified by:
`containsValue` in interface `Map`

Parameters:
`value` - value whose presence in this map is to be tested.

Returns:
`true` if this map maps one or more keys to the specified value.

Throws:
`UnsupportedOperationException` - indicating this method is not supported

---

get

`Object` **`get`**`(Object key)`

Retrieves the object from the cache at the given key.

Whether or not a copy of the object is returned is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if a value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.get(Object)` for additional specification details.

Specified by:
`get` in interface `Map`

Parameters:
`key` - The entry to fetch

Returns:
the value, `null`, `SerializedValue` or `Tuple`

Throws:
`IllegalArgumentException` - if key is `null`
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:
`containsKey(Object)`, `getForUpdate(Object)`, `CopyMode`, `ObjectMap.get(Object)`

---

put

`Object` **`put`**`(Object key,`
`Object value)`

Puts the Object value into the cache at location represented by key.

The values will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the `ObjectMap.setPutMode(PutMode)` property:

ObjectMap.PutMode.INSERTUPDATE	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
ObjectMap.PutMode.UPSERT	The values are put into the map using the specification of the ObjectMap.upsertAll(LinkedHashMap).

Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

The return value is a `SerializedValue` when using the CopyMode.COPY_TO_BYTES_RAW CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See ObjectMap.put(Object, Object) for additional specification details.

Specified by:

put in interface Map

Parameters:

`key` - The entry to put into the map

`value` - The value to put into the map using the key

Returns:

If ObjectMap.PutMode.INSERTUPDATE is set, return the previous value in this transaction. If ObjectMap.PutMode.UPSERT is set, the return value is null.

Throws:

IllegalArgumentException - if key is `null`, or if the map does not allow `null` values and value is `null`

ObjectGridRuntimeException - if an error occurs during processing

See Also:

CopyMode, ObjectMap.put(Object, Object)

---

remove

Object **remove**(Object key)

Removes the Object value from the cache represented by key.

This removal will be pushed down to the BackingMap/Loader at commit time. If the key cannot be found in the map, a `null` value will be returned.

The return value is a `SerializedValue` when using the CopyMode.COPY_TO_BYTES_RAW CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See ObjectMap.remove(Object) for additional specification details.

Specified by:

remove in interface Map

Parameters:

`key` - The entry to remove

Returns:

the current value at invocation time

Throws:

IllegalArgumentException - if key is `null`

ObjectGridRuntimeException - if an error occurs during processing

See Also:

ObjectMap.remove(Object)

---

putAll

`void` **`putAll`**`(Map map)`

Puts each of the Object values into the cache at location represented by the corresponding key contained in the Map.

The value will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the ObjectMap.setPutMode(PutMode) property:

ObjectMap.PutMode.INSERTUPDATE	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
ObjectMap.PutMode.UPSERT	The value is put into the map using the specification of the ObjectMap.upsert(Object, Object).

Whether or not a copy of the objects contained in the map is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

An existing Map object will be passed in to use for obtaining the keys and values to be inserted or updated into the existing Map.

See ObjectMap.putAll(Map) for additional specification details.

Specified by:

putAll in interface Map

Parameters:

`map` - The key/values to be put into the map.

Throws:

IllegalArgumentException - if map is `null` or contains a `null` key or if `null` values are not allowed and map contains a `null` value.

ObjectGridRuntimeException - if an error occurs during processing

See Also:

put(Object, Object), ObjectMap.putAll(Map)

---

clear

`void` **`clear`**`()`

Clear all keys from the Map.

This method is an auto-commit call, so a session should not be explicitly begun or committed when calling clear on the Map.

Specified by:

clear in interface Map

Throws:

ObjectGridRuntimeException - if an error occurs during processing

Since:

keySet

`Set` **`keySet`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:
`keySet` in interface `Map`

Returns:
a set view of the keys contained in this map.
Throws:
`UnsupportedOperationException` - indicating this method is not supported

values

`Collection` **`values`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:
`values` in interface `Map`

Returns:
a collection view of the values contained in this map.
Throws:
`UnsupportedOperationException` - indicating this method is not supported

entrySet

`Set` **`entrySet`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:
`entrySet` in interface `Map`

Returns:
a set view of the mappings contained in this map.
Throws:
`UnsupportedOperationException` - indicating this method is not supported

touch

`void` **`touch`**`(``Object` `key)`

Updates the last access time in the BackingMap without retrieving the value to the JavaMap.

The last access time is updated during commit. If the key does not exist in the BackingMap, a TransactionException will be returned during commit processing.

See `ObjectMap.touch(Object)` for additional specification details.

Parameters:
`key` - key to be touched
Throws:

IllegalArgumentException - if key is `null`
ObjectGridRuntimeException - if an error occurs during processing

See Also:

ObjectMap.touch(Object)

---

setCopyMode

```
void setCopyMode(CopyMode copyMode,
                 Class valueInterface)
                 throws ObjectGridRuntimeException
```

Allows the CopyMode for the Map to be overridden on this map on this session only. This method allows an application to use an optimal CopyMode TRANSACTION by TRANSACTION as its needs dictate. The CopyMode cannot be changed during a transaction. There must be no active transaction when this method is called.

Parameters:

`copyMode` - must be one of the final static variables defined in `CopyMode`. See `CopyMode` class for an explanation of each mode and how the valueInterface is used for `CopyMode.COPY_ON_WRITE` .
`valueInterface` - the value interface Class object. Specify null in version 7.1 and later.

Throws:

IllegalArgumentException - if copyMode is `null` or COPY_ON_WRITE CopyMode is specified and the required value interface parameter is `null`
ObjectGridRuntimeException - if a transaction is active and this map has already been used in the transaction or an error occurs during processing

See Also:

BackingMap.setCopyMode(CopyMode, Class), CopyMode, ObjectMap.setCopyMode(CopyMode, Class)

---

clearCopyMode

```
void clearCopyMode()
                 throws ObjectGridRuntimeException
```

Resets the CopyMode back to the one in the BackingMap.

This method is used to reverse a previous setCopyMode method call for this JavaMap. This method can only be called when no transaction is active on the associated session.

Throws:

ObjectGridRuntimeException - if a transaction is active and this map has already been used in the transaction or an error occurs during processing

See Also:

setCopyMode(CopyMode, Class), ObjectMap.clearCopyMode()

---

getNextKey

```
Object getNextKey(long timeout)
```

Retrieves a key off the map in first-in-first-out (FIFO) insert order. The entry is locked by the session such that other calls to getNextKey will not return the same key. The key can be used to remove or manipulate the value although leaving the entry will result in the key remaining at the beginning of the queue. This order is optimized for performance and is not guaranteed especially across partitions or in highly concurrent environments.

The return value is a `SerializedKey` when OutputFormat.RAW is set for the keys. The default key output format for maps that are associated with a KeySerializerPlugin is OutputFormat.RAW. The SerializedKey allows access to the value in its serialized form, or

its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getNextKey(long)` for additional specification details.

Parameters:
`timeout` - The period of time to wait for an entry to become available on the queue.
Returns:
The next available key in the map.
See Also:
`ObjectMap.getNextKey(long)`

---

getEntityMetadata

`com.ibm.websphere.projector.md.EntityMetadata` **`getEntityMetadata`**`()`

Retrieve the metadata for the entity associated with this map.

Returns:
the EntityMetadata if an entity is associated with this map or null if there is no entity associated with this map.
Since:
WAS XD 6.1

---

com.ibm.websphere.objectgrid

Interface IObjectGridException

All Known Implementing Classes:

CacheEntryException, CannotGenerateCredentialException, ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException, com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException, ConnectException, ContinuousQueryAttributePathException, ContinuousQueryException, ContinuousQueryGetValueException, ContinuousQueryIncompatibleDuplicateException, DuplicateKeyException, ExpiredCredentialException, InvalidCredentialException, KeyNotFoundException, LoaderException, LockDeadlockException, LockException, LockInternalFailureException, LockTimeoutException, NoActiveTransactionException, ObjectGridConfigurationException, ObjectGridException, ObjectGridRuntimeException, ObjectGridSecurityException, ReadOnlyException, ReplicationVotedToRollbackTransactionException, SessionNotReentrantException, TransactionAffinityException, TransactionAlreadyActiveException, TransactionCallbackException, TransactionException, TransactionQuiesceException, TransactionTimeoutException, UnavailableServiceException, UndefinedMapException

---

```
public interface IObjectGridException
```

This interface is used to ensure JDK 1.4 Throwable chaining behavior for all exceptions thrown by ObjectGrid even when an earlier JDK is used (e.g. JDK 1.3.1).

Since:

WAS XD 6.0.1, XC10

Method Summary

Throwable	**getCause**() Provides JDK 1.4 Throwable.getCause() behavior.
Throwable	**initCause**(Throwable cause) Provides JDK 1.4 Throwable.initCause() behavior.

Method Detail

getCause

Throwable **getCause**()

Provides JDK 1.4 Throwable.getCause() behavior.

Returns the cause of this throwable or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this throwable to get thrown.)

This implementation returns the cause that was supplied via one of the constructors requiring a `Throwable`, or that was set after creation with the `initCause(Throwable)` method. While it is typically unnecessary to override this method, a subclass can override it to return a cause set by some other means. This is appropriate for a "legacy chained throwable" that predates the addition of chained exceptions to `Throwable`. Note that it is *not* necessary to override any of the `PrintStackTrace` methods, all of which invoke the `getCause` method to determine the cause of a throwable.

Returns:
the cause of this throwable or `null` if the cause is nonexistent or unknown.
See Also:
initCause(Throwable)

---

initCause

Throwable **initCause**(Throwable cause)
throws IllegalArgumentException,
IllegalStateException

Provides JDK 1.4 Throwable.initCause() behavior.

Initializes the *cause* of this throwable to the specified value. (The cause is the throwable that caused this throwable to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the throwable. If this throwable was created with `Throwable(Throwable)` or `Throwable(String,Throwable)`, this method cannot be called even once.

Parameters:
`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)
Returns:
a reference to this `Throwable` instance.
Throws:
IllegalArgumentException - if `cause` is this throwable. (A throwable cannot be its own cause.)
IllegalStateException - if this throwable was created with `Throwable(Throwable)` or `Throwable(String,Throwable)`, or this method has already been called on this throwable.

---

com.ibm.websphere.objectgrid

Class DuplicateKeyException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.DuplicateKeyException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class DuplicateKeyException
extends ObjectGridException
```

A DuplicateKeyException exception is thrown if a key cannot be inserted into a BackingMap because an object with the same key already exists.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

DuplicateKeyException() Constructs a new DuplicateKeyException with `null` as its detail message.
DuplicateKeyException(String message) Constructs a new DuplicateKeyException with the specified detail message.
DuplicateKeyException(String message, Throwable cause) Constructs a new DuplicateKeyException with the specified detail message and cause.
DuplicateKeyException(Throwable cause) Constructs a new DuplicateKeyException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

DuplicateKeyException

```
public DuplicateKeyException()
```

Constructs a new DuplicateKeyException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

[ObjectGridException.initCause(Throwable)]

DuplicateKeyException

```
public DuplicateKeyException(String message)
```

Constructs a new DuplicateKeyException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

[ObjectGridException.initCause(Throwable)], [Throwable.getMessage()]

DuplicateKeyException

```
public DuplicateKeyException(String message,
                             Throwable cause)
```

Constructs a new DuplicateKeyException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this DuplicateKeyException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

[ObjectGridException.getCause()], [Throwable.getMessage()]

DuplicateKeyException

```
public DuplicateKeyException(Throwable cause)
```

Constructs a new DuplicateKeyException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for DuplicateKeyExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Class CopyMode

java.lang.Object

  com.ibm.websphere.objectgrid.CopyMode

```
public final class CopyMode
extends Object
```

This class is used to define the "copy" mode when the `setCopyMode` method of the `BackingMap` interface is used. The application is expected to pass one of the final static variables that are defined in this class to the `setCopyMode` method.

Since:

WAS XD 6.0, XC10

See Also:

`BackingMap.setCopyMode(CopyMode, Class)`, `ObjectTransformer.copyValue(Object)`

Field Summary

static CopyMode	**COPY_ON_READ** The COPY_ON_READ mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when a transaction is committed.
static CopyMode	**COPY_ON_READ_AND_COMMIT** The COPY_ON_READ_AND_COMMIT mode is the default mode.
static CopyMode	**COPY_ON_WRITE** The COPY_ON_WRITE mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when `ObjectMap.get` is called for the first time by a transaction for a given key.

[Mode](#)	
static [CopyMode](#)	**COPY_TO_BYTES** The COPY_TO_BYTES mode is similar to the COPY_ON_READ_AND_COMMIT mode in that it ensures that an application never has a reference to the value object that is in the BackingMap.
static [CopyMode](#)	**COPY_TO_BYTES_RAW** When set, all ObjectMap APIs that return a `SerializedValue` rather than the original Java Object, allowing access to the serialized form of the data, preventing inflation of object into Java Object form.
static [CopyMode](#)	**NO_COPY** The NO_COPY mode allows an application to promise that it will never modify a value object obtained using an ObjectMap.get method in exchange for performance improvements.

Method Summary

boolean	**isBytes**() Is the copy mode one of the copy modes that indicate copy to bytes?
[String](#)	**toString**() Returns a string representation of the `CopyMode`.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

COPY_ON_READ_AND_COMMIT

`public static final CopyMode` **`COPY_ON_READ_AND_COMMIT`**

The COPY_ON_READ_AND_COMMIT mode is the default mode. This mode ensures that an application never has a reference to the value object that is in the BackingMap, and instead the application is always working with a copy of the value that is in the BackingMap. The copy ensures the application can never inadvertently corrupt the data that is cached in the BackingMap. When an application transaction calls an `ObjectMap.get` method for a given key, and it is the first access of the ObjectMap entry for that key, a copy of the value is returned. When the transaction is committed, any changes the application committed are copied to the BackingMap to ensure that the application does not have reference to the committed value in the BackingMap.

COPY_ON_READ

`public static final CopyMode` **`COPY_ON_READ`**

The COPY_ON_READ mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when a transaction is committed. To preserve integrity of BackingMap data, the application promises to destroy every reference it has to an entry once the transaction is committed. This mode results in a `ObjectMap.get` method returning a copy of the value rather than a reference to the value to ensure that changes made by the application to the value does not affect the BackingMap value until the transaction is committed. However, when the transaction does commit, a copy of changes is not made. Instead, the reference to the copy that was returned by ObjectMap.get is stored in the BackingMap. This is the reason the application must agree to destroy all map entry references once the transaction is committed. If application fails to keep its promise, the application could cause the data cached in BackingMap to become corrupted. If an application is using this mode and it is having problems, then switch to the COPY_ON_READ_AND_COMMIT mode to see if the problem still exists. If the problem goes away, then more than likely the application is failing to destroy all of its references after the transaction has committed.

COPY_ON_WRITE

`public static final CopyMode` **`COPY_ON_WRITE`**

The COPY_ON_WRITE mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when `ObjectMap.get` is called for the first time by a transaction for a given key. Instead, the ObjectMap.get method returns a proxy to the value rather than a direct reference to the value object itself. The proxy ensures that a copy of the value is not made unless the application calls a set method on the value interface that is passed on the `BackingMap.setCopyMode(CopyMode, Class)` method. Thus, the proxy provides a "copy on write" implementation. When a transaction commits, the `BackingMap` examines the proxy to determine if any copy was made as a result of a set method being called. If a copy was made, then the reference to that copy is stored in the BackingMap. The big advantage of this mode is a value is never copied on read or at commit when the transaction never calls a set method to mutate the value.

See Also:

`BackingMap.setCopyMode(CopyMode, Class)`

NO_COPY

`public static final CopyMode` **`NO_COPY`**

The NO_COPY mode allows an application to promise that it will never modify a value

object obtained using an ObjectMap.get method in exchange for performance improvements. If this mode is used, no copy of the value is ever made. If the application breaks its promise and does modify values, then data in the BackingMap will be corrupted. This mode is primarily useful for read only maps where data is never modified by the application. If the application is using this mode and it is having problems, then switch to COPY_ON_READ_AND_COMMIT mode to see if the problem still exists. If the problem goes away, then more than likely the application is not keeping its promise and is modifying the value returned by ObjectMap.get method (either during transaction or after transaction has committed).

COPY_TO_BYTES

`public static final CopyMode COPY_TO_BYTES`

The COPY_TO_BYTES mode is similar to the COPY_ON_READ_AND_COMMIT mode in that it ensures that an application never has a reference to the value object that is in the BackingMap. The value that the application works with is a newly inflated version of the serialized version that is in the BackingMap. The copy ensures the application can never inadvertently corrupt the data that is cached in the BackingMap since a byte form of the value is what is stored in the BackingMap instead of the Object form.

A copy of the value is returned when an application transaction calls an `ObjectMap.get` method for a given key, and it is the first time that the ObjectMap entry is accessed for that key. When the transaction is committed, any changes the application committed are copied to bytes in the BackingMap to ensure that the application does not have reference to the committed value in the BackingMap.

Since:
7.0

COPY_TO_BYTES_RAW

`public static final CopyMode COPY_TO_BYTES_RAW`

When set, all ObjectMap APIs that return a `SerializedValue` rather than the original Java Object, allowing access to the serialized form of the data, preventing inflation of object into Java Object form.

Since:
7.1.1
See Also:
`ValueDataSerializer`

Method Detail

toString

`public String toString()`

Returns a string representation of the `CopyMode`.

Overrides:
`toString` in class `Object`

Returns:
a string representation of the `CopyMode`.

---

isBytes

`public boolean isBytes()`

Is the copy mode one of the copy modes that indicate copy to bytes?

Returns:

boolean indicating if copy mode is one of COPY_TO_BYTES or COPY_TO_BYTES_RAW.

---

com.ibm.websphere.objectgrid

Class ConnectException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.ConnectException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **ConnectException**
extends ObjectGridException

This exception is used to indicate that the client was unable to connect to the server

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Field Summary

static int	**BAD_CONFIGURATION** This failure code indicates the provided configuration was corrupt.
static int	**CONNECTION_REFUSED** This failure code indicates that the server may not be available.
static int	**FAILED_SECURITY** This failure code indicates the a failure to authenticate.
s	

static int	**SERVER_DEFINITION_NOT_FOUND** This failure code indicates the definition of cluster cannot be accessed.
static int	**UNKNOWN** This failure code indicates the reason for the connect failure is unknown.

Constructor Summary

ConnectException() Constructs a new ConnectException with `null` as its detail message.
ConnectException(String message) Constructs a new ConnectException with the specified detail message.
ConnectException(String message, int failureCode) Constructs a new ConnectException with the specified detail message.
ConnectException(String message, Throwable cause) Constructs a new ConnectException with the specified detail message and cause.
ConnectException(String message, Throwable cause, int failureCode) Constructs a new ConnectException with the specified detail message and cause.
ConnectException(Throwable cause) Constructs a new ConnectException with a specified cause.

Method Summary

int	**getFailureCode**() Returns the failure code that was set by one of the constructors that accepts a failure code, or `UNKNOWN` if one of the other constructors was called.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

BAD_CONFIGURATION

public static final int **BAD_CONFIGURATION**

This failure code indicates the provided configuration was corrupt.

See Also:
Constant Field Values

UNKNOWN

`public static final int` **`UNKNOWN`**

This failure code indicates the reason for the connect failure is unknown.

See Also:
Constant Field Values

FAILED_SECURITY

`public static final int` **`FAILED_SECURITY`**

This failure code indicates the a failure to authenticate.

See Also:
Constant Field Values

CONNECTION_REFUSED

`public static final int` **`CONNECTION_REFUSED`**

This failure code indicates that the server may not be available.

See Also:
Constant Field Values

SERVER_DEFINITION_NOT_FOUND

`public static final int` **`SERVER_DEFINITION_NOT_FOUND`**

This failure code indicates the definition of cluster cannot be accessed.

See Also:
Constant Field Values

Constructor Detail

ConnectException

`public` **`ConnectException`**`()`

Constructs a new ConnectException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method. In addition the failure code is initialized to `UNKNOWN`.

See Also:
`ObjectGridException.initCause(Throwable)`, `UNKNOWN`

ConnectException

`public` **`ConnectException`**`(`String message`)`

Constructs a new ConnectException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method. In addition the failure code is initialized to `UNKNOWN`.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`, `UNKNOWN`

ConnectException

```
public ConnectException(String message,
                        int failureCode)
```

Constructs a new ConnectException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`failureCode` - the failure code which should be one of the constants of this exception class.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`, `getFailureCode()`

ConnectException

```
public ConnectException(Throwable cause)
```

Constructs a new ConnectException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ConnectExceptions that are little more than wrappers for other throwables. The failure code is initialized to `UNKNOWN`.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`, `UNKNOWN`

ConnectException

```
public ConnectException(String message,
                        Throwable cause)
```

Constructs a new ConnectException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ConnectException's detail message. The failure code is initialized to `UNKNOWN`.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

ConnectException

```
public ConnectException(String message,
                        Throwable cause,
                        int failureCode)
```

Constructs a new ConnectException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ConnectException's detail message. The failure code is initialized to `UNKNOWN`.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
`failureCode` - the failure code which should be one of the constants of this exception class.

See Also:

ObjectGridException.getCause(), Throwable.getMessage(), getFailureCode()

Method Detail

getFailureCode

```
public int getFailureCode()
```

Returns the failure code that was set by one of the constructors that accepts a failure code, or `UNKNOWN` if one of the other constructors was called.

Returns:

the failure code. One of the constants of this exception class.

---

com.ibm.websphere.objectgrid

Class ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.TransactionCallbackException
          com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException
            com.ibm.websphere.objectgrid.ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException
extends com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException
```

This exception is thrown when a method call to the Client/Server TransactionCallback detects the user is attempting to perform a write against multiple maps in different Map Sets, Partition Sets or Replication groups. This is not allowed.

Since:

WAS XD 6.0.1, XC10

See Also:

TransactionCallback, Serialized Form

Constructor Summary

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException() Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with `null` as its detail message.
ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException(String message) Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message.
ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException(String message, Throwable cause) Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message and cause.
ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException(Throwable cause) Constructs a new

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

public **ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException**()

Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

public **ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException**(String message)

Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

public **ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException**(String message, Throwable cause)

Constructs a new

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

`public` **`ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException`**`(Throwable cause)`

Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Interface ClientReplicableMap

All Known Subinterfaces:

BackingMap

Deprecated. *The client replicated map function is deprecated in version 8.6. Use the `ContinuousQueryManager` function.*

```
public interface ClientReplicableMap
```

This interface represents a replicable client map. A replicable client map can be a continuous replica or a snapshot replica of the server map.

If the client is a continuous replica of the server map, the data in the server will be replicated to the client continuously in an asynchronous manner.

If the client is a snapshot replica of the server map, a snapshot on the data in the server will be taken and the snapshot will be replicated to the client in an asynchronous manner. A snapshot replication is a one-time replication.

A ReplicationMapListener can be used to listen for the data changes as well as the replication lifecycle events.

Since:
WAS XD 6.1, XC10

See Also:
ReplicationMapListener

Nested Class Summary

static class	**ClientReplicableMap.Mode** **Deprecated.** Client Replication mode

Field Summary

static Clie	

[IClientReplicableMap.Mode](#)	**CONTINUOUS_REPLICATION** **Deprecated.** Full replication mode.
[IClientReplicableMap.Mode](#)	**NONE** **Deprecated.** No replication mode, aka normal mode.
[IClientReplicableMap.Mode](#)	**SNAPSHOT_REPLICATION** **Deprecated.** Snapshot replication mode.

[o d e](#)	

Method Summary

void	**disableClientReplication**() **Deprecated.** Disables the replication for this client.
void	**enableClientReplication**(ClientReplicableMap.Mode mode, int[] partitions, ReplicationMapListener listener) **Deprecated.** Make the client map a replica of the server side map.
ClientReplicableMap.Mode	**getReplicationMode**() **Deprecated.** Returns the current replication mode

Field Detail

NONE

static final ClientReplicableMap.Mode **NONE**

Deprecated.
No replication mode, aka normal mode.

CONTINUOUS_REPLICATION

static final ClientReplicableMap.Mode **CONTINUOUS_REPLICATION**

Deprecated.
Full replication mode. Data in the server map will be replicated to the client continuously.

SNAPSHOT_REPLICATION

static final ClientReplicableMap.Mode **SNAPSHOT_REPLICATION**

Deprecated.

Snapshot replication mode. A snapshot on the data in the server will be taken and the snapshot will be replicated to the client. A snapshot replication is a one-time replication.

Method Detail

enableClientReplication

```
void enableClientReplication(ClientReplicableMap.Mode mode,
                             int[] partitions,
                             ReplicationMapListener listener)
                             throws ObjectGridException
```

Deprecated.
Make the client map a replica of the server side map.

When security is enabled, this method requires a `ServerMapPermission` with action "replicate". Refer to `ServerMapPermission` for more permission details.

Required Client Permission: `ServerMapPermission.REPLICATE`

Parameters:

`mode` - The replication mode.
`partitions` - The array of partition IDs represent which partitions the data should be replicated from. If the value is null or an empty array, it indicates the data should be replicated from all partitions.
`listener` - a listener to receive client replication events

Throws:

`IllegalArgumentException` - if mode is not `CONTINUOUS_REPLICATION` or `SNAPSHOT_REPLICATION` or the map isn't currently in the mode specified or is not in `NONE` mode
`IllegalStateException` - if this method is invoked on a map other than a client map
`ObjectGridException` - if an error occurs during processing this request

See Also:

`ReplicationMapListener`, `CONTINUOUS_REPLICATION`, `SNAPSHOT_REPLICATION`, `getReplicationMode()`, `BackingMap.CLIENT`

---

getReplicationMode

```
ClientReplicableMap.Mode getReplicationMode()
```

Deprecated.
Returns the current replication mode

Returns:

the replication mode

See Also:

`NONE`, `CONTINUOUS_REPLICATION`, `SNAPSHOT_REPLICATION`

---

disableClientReplication

```
void disableClientReplication()
                            throws ObjectGridException
```

Deprecated.
Disables the replication for this client. If it is not in a replication mode, this method will be a no-op.

When security is enabled, this method requires a `ServerMapPermission` with action "replicate". Refer to `ServerMapPermission` for more permission details.

Throws:

IllegalStateException - if this method is invoked on a map other than a client map
ObjectGridException - if an error occurs during processing this request

See Also:

BackingMap.CLIENT

---

com.ibm.websphere.objectgrid

Class ClientReplicableMap.Mode

java.lang.Object

com.ibm.websphere.objectgrid.ClientReplicableMap.Mode

Enclosing interface:

ClientReplicableMap

```
public static final class ClientReplicableMap.Mode
extends Object
```

Client Replication mode

Method Summary

String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Method Detail

toString

```
public String toString()
```

Overrides:

toString in class Object

com.ibm.websphere.objectgrid

Interface ClientClusterContext

public interface **ClientClusterContext**

This interface is a context to represent which cluster/domain the client connected to using one of the `ObjectGridManager.connect` methods. An instance of this interface is used to retrieve client `ObjectGrid` instances and for performing admin operations against an ObjectGrid cluster/domain or its servers.

Since:
WAS XD 6.0.1, XC10

See Also:
ObjectGridManager

Method Summary

ClientProperties	**getClientProperties**(String objectGridName) Retrieve the ClientProperties object for this ClientClusterContext for the specified ObjectGrid name.
String	**getClusterName**() Gets the name of the domain to which the client is connected
void	**setClientProperties**(String objectGridName, URL url) Sets the ClientProperties properties for the selected ObjectGrid using the specified client properies file.

Method Detail

getClusterName

String **getClusterName**()

Gets the name of the domain to which the client is connected

Returns:
the name of the domain this context is connected to

---

getClientProperties

`ClientProperties` **`getClientProperties`**`(String objectGridName)`

Retrieve the ClientProperties object for this ClientClusterContext for the specified ObjectGrid name. A ClientProperties is scoped to this ClientClusterContext and a single ObjectGrid.

Parameters:
`objectGridName` - the name of ObjectGrid
Returns:
the ClientProperties instance for this ObjectGrid.
Since:
WAS XD 6.1.0.3

---

setClientProperties

```
void setClientProperties(String objectGridName,
                         URL url)
```

Sets the ClientProperties properties for the selected ObjectGrid using the specified client properies file.

To further adjust the client properties, call the `getClientProperties(String)` method.

Parameters:
`objectGridName` - the name of ObjectGrid to apply the ClientProperties to.
`url` - the URL where the client properties file can be located.
Since:
WAS XD 6.1.0.3

---

com.ibm.websphere.objectgrid

Interface CatalogDomainManager

public interface **CatalogDomainManager**

Provides access to catalog domain configuration information for the current environment.

When running in a WebSphere Application Server profile augmented with WebSphere eXtreme Scale, the CatalogDomainManager returns the catalog service domain configuration information that is configured in the administration console.

Since:
8.5, XC10

See Also:
ObjectGridManager.getCatalogDomainManager()

Method Summary

CatalogDomainInfo	**getDefaultDomainInfo**() Retrieve the default, configured CatalogDomainInfo.
CatalogDomainInfo	**getDomainInfo**(String domainId) Retrieve the specified CatalogDomainInfo for the specified domainId.
Coll	

Collection<CatalogDomainInfo>	**getDomainInfos**() Retrieve all configured CatalogDomainInfo objects.

Method Detail

getDefaultDomainInfo

CatalogDomainInfo **getDefaultDomainInfo**()

Retrieve the default, configured CatalogDomainInfo.

Returns:
the default CatalogDomainInfo, or null if not available.

---

getDomainInfo

CatalogDomainInfo **getDomainInfo**(String domainId)

Retrieve the specified CatalogDomainInfo for the specified domainId.

Parameters:
`domainId` - the domain identifier.
Returns:
the CatalogDomainInfo if found, or null.

---

getDomainInfos

Collection<CatalogDomainInfo> **getDomainInfos**()

Retrieve all configured CatalogDomainInfo objects.

Returns:
a collection of CatalogDomainInfo

---

com.ibm.websphere.objectgrid

Interface CatalogDomainInfo

public interface **CatalogDomainInfo**

Identifies the configuration attributes of a catalog service domain.

Since:
8.5, XC10

See Also:
CatalogDomainManager

Method Summary

String	**getClientCatalogServerEndpoints**() Retrieve the catalog server endpoints used to connect a client to the remote catalog service domain.
ClientSecurityConfiguration	**getClientSecurityConfiguration**() Retrieve the ClientSecurityConfiguration for the domain.
String	**getDomainId**() Retrieve the identifier of the domain as specified by the catalog service domain.

Method Detail

getDomainId

String **getDomainId**()

Retrieve the identifier of the domain as specified by the catalog service domain.

Note: This is different than the name of the domain which is specified when starting the catalog services.

Returns:
the identifier of the domain.

---

getClientCatalogServerEndpoints

String **getClientCatalogServerEndpoints**()

Retrieve the catalog server endpoints used to connect a client to the remote catalog service domain.

The catalog service endpoints are used with the ObjectGridManager.connect(String, com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration, java.net.URL) method to connect to a catalog service domain.

Returns:
the catalog service endpoints.

---

getClientSecurityConfiguration

ClientSecurityConfiguration **getClientSecurityConfiguration**()

Retrieve the ClientSecurityConfiguration for the domain.

The ClientSecurityConfiguration are used with the ObjectGridManager.connect(String, com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration, java.net.URL) method to connect to a catalog service domain.

Returns:
the ClientSecurityConfiguration or null if security is not configured.

---

com.ibm.websphere.objectgrid

Interface BackingMap

All Superinterfaces:

ClientReplicableMap

```
public interface BackingMap
extends ClientReplicableMap
```

This is the public interface to the BackingMap. It is returned when a new Map is defined on the ObjectGrid. It allows the Map to be customized with various plug-ins or by setting properties. The defaults are:

- No external `Evictor`, but an internal time-based evictor is provided by default
- No `Loader`
- No `EventListeners`
- No `MapIndexPlugins`
- An internal `ObjectTransformer`
- An internal `OptimisticCallback`
- Key is not copied
- A value `CopyMode` of `CopyMode.COPY_ON_READ_AND_COMMIT`
- A `LockStrategy` of `LockStrategy.OPTIMISTIC`
- A default lock timeout
- `null` values are supported
- A default number of buckets
- A default number of lock buckets
- Synchronous preload
- Read/write map by default
- A TimeToLive of 0 (indicating unlimited time)
- A TtlEvictor type of `TTLType.NONE`
- Write-behind updates is disabled
- Time-based database updates are disabled
- Eviction triggers are not set

Since:

WAS XD 6.0, XC10

Nested Class Summary

Nested classes/interfaces inherited from interface com.ibm.websphere.objectgrid.ClientReplicableMap
ClientReplicableMap.Mode

Field Summary

static	**CLIENT**

static int	Constant used to indicate this map is a client to a server map
static int	**DEFAULT_LOCK_TIMEOUT** Default lock timeout used if `setLockTimeout(int)` is not invoked.
static int	**DEFAULT_NUMBER_OF_BUCKETS** **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the [isNearCacheEnabled()] flag to disable the near cache in the ObjectGrid configuration XML file.*
static int	**DEFAULT_NUMBER_OF_LOCK_BUCKETS** Default number of lock buckets used if `setNumberOfLockBuckets(int)` is not invoked.
static String	**EVICTIONTRIGGER_MEMORY_USAGE_THRESHOLD** The eviction trigger string constant to enable memory based eviction using memory usage threshold provided by the `java.lang.management.MemoryPoolMXBean`.
static int	**LOCAL** Constant used to indicate this map is not a distributed map.
static int	**SERVER** Constant used to indicate this map is a server map.

Fields inherited from interface com.ibm.websphere.objectgrid.ClientReplicableMap
CONTINUOUS_REPLICATION, NONE, SNAPSHOT_REPLICATION

Method Summary

void	**addMapEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener eventListener) Adds an EventListener to this BackingMap.
void	**addMapEventListener**(MapEventListener eventListener) **Deprecated.** *This method is deprecated in version 7.1.1, use the addMapEventListener(EventListener) method.*
void	**addMapIndexPlugin**(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index) Adds an MapIndexPlugin to this Map.
void	**createDynamicIndex**(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index, com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicIndexCallback) Creates a dynamic index on the BackingMap.
void	**createDynamicIndex**(String name, boolean isRangeIndex, String attributeName, com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicIndexCallback) Creates a dynamic index on the BackingMap.
boolean	**getCopyKey**() Gets whether keys are copied for this BackingMap.
CopyMode	**getCopyMode**() Gets the CopyMode being used by this BackingMap.
com.ibm.websphere.projector.	**getEntityMetadata**() Retreive the metadata for the entity associated with this backing map.

md.EntityMetadata	
String	**getEvictionTriggers**() Returns the types of additional eviction triggers.
Evictor	**getEvictor**() Gets the `Evictor` being used by this BackingMap.
com.ibm.websphere.objectgrid.OutputFormat	**getKeyOutputFormat**() Retrieves the data format for all data access APIs that return cache keys.

com.ibm.websphere.objectgrid.plugins.Loader	**getLoader**() Gets the Loader being used by this BackingMap.
LockStrategy	**getLockStrategy**() Gets the LockStrategy object being used by this BackingMap.
int	**getLockTimeout**() Gets the lock timeout value used by the lock manager for this BackingMap.
List	**getMapEventListeners**() Gets the current list of EventListenerS.
List	**getMapIndexPlugins**() Returns the current list of MapIndexPlugin objects for this BackingMap.
String	**getMapSetName**() Retrieves the name of the MapSet that this BackingMap is currently associated with.

[ng](#)	
int	**getMapType**() Returns the type of BackingMap.
[String](#)	**getName**() Gets the name of the `BackingMap`.
boolean	**getNullValuesSupported**() Gets whether this `BackingMap` supports `null` values or not.
int	**getNumberOfBuckets**() **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the [isNearCacheEnabled()](#) flag to disable the near cache in the ObjectGrid configuration XML file.*
int	**getNumberOfLockBuckets**() Gets the number of lock buckets defined for the hash map used by lock manager for this backing map.
[ObjectGrid](#)	**getObjectGrid**() Gets the `ObjectGrid` that owns this `BackingMap`.
com.ibm.websphere.objectgrid.plug	**getObjectTransformer**() Gets the `ObjectTransformer` object being used by this `BackingMap` and/or `Loader`.

ins.ObjectTransformer	
com.ibm.websphere.objectgrid.plugins.OptimisticCallback	**getOptimisticCallback**() Gets the OptimisticCallback being used by this BackingMap and/or Loader or null if the LockStrategy is not optimistic.

int	**getPartitionId**() Gets the partition identifier being used by this BackingMap.
PartitionManager	**getPartitionManager**() Allows access to the PartitionManager that is defined for this BackingMap.
boolean	**getPreLoadMode**() Returns whether this BackingMap will be asynchronously preloaded or not if a Loader is set.
boolean	**getReadOnly**() Retrieves the map type.
com.ibm.websphere.objectgrid.plugins.	**getSerializerAccessor**() Retrieve the SerializerAccessor for this map.

io.SerializerAccessor	
com.ibm.websphere.objectgrid.plugins.BackingMapLifecycl	**getState**() Retrieve the current life cycle state of this map.

eListener.State	
com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig	**getTimeBasedDBUpdateConfig**() Get the time-based database update configuration object.
int	**getTimeToLive**() Gets the number of seconds for an entry to live.
TTL	**getTtlEvictorType**()

Type	
	Gets how expiration time of a `BackingMap` entry is computed.
com.ibm.websphere.objectgrid.OutputFormat	**getValueOutputFormat**() Retrieves the data output format for all data access APIs that return cache values.
String	**getWriteBehind**() Get the write-behind parameter.
boolean	**isNearCacheEnabled**() If true, the client near cache is enabled for supported configurations.
boolean	**isNearCacheInvalidationEnabled**() If true, clients with local caches are automatically invalidated when the data grid map is updated.
boole	**isNearCacheLastAccessTTLSyncEnabled**() If true, clients automatically send time-to-live access information to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.

an	
void	**removeDynamicIndex**(String name) Removes a dynamic index on the BackingMap.
void	**removeMapEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener eventListener) Removes an EventListener from this BackingMap.
void	**removeMapEventListener**(MapEventListener eventListener) Provided for compatibility with old releases, use the removeMapEventListener(EventListener) method.
void	**setCopyKey**(boolean copy) Sets whether or not the key needs to be copied when a map entry is created.
void	**setCopyMode**(CopyMode mode, Class valueInterface) Sets the CopyMode.
void	**setEvictionTriggers**(String evictionTriggers) Sets the types of additional eviction triggers, all evictors for the backing map will use the provided set of triggers.
void	**setEvictor**(Evictor e) Associates an Evictor with this BackingMap.
void	**setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data output format for all data access APIs that return cache keys.
void	**setLoader**(com.ibm.websphere.objectgrid.plugins.Loader loader) Associates a Loader with this BackingMap.
void	**setLockStrategy**(LockStrategy lockStrategy) Sets the LockStrategy.
void	**setLockTimeout**(int seconds) Sets the lock timeout used by the lock manager for this BackingMap.
void	**setMapEventListeners**(List eventListenerList) **Deprecated.** *This method is deprecated in version 7.1.1. Use the addMapEventListener(EventListener) or removeMapEventListener(EventListener) methods. Plugins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*
void	**setMapIndexPlugins**(List indexList) Sets the list of MapIndexPlugin objects for this BackingMap.
v	

void	**setNullValuesSupported**(boolean nullValuesSupported) Sets whether this BackingMap supports null values.
void	**setNumberOfBuckets**(int numBuckets) **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the isNearCacheEnabled() flag to disable the near cache in the ObjectGrid configuration XML file.*
void	**setNumberOfLockBuckets**(int numBuckets) Sets the number of lock buckets used by the lock manager for this BackingMap.
void	**setObjectTransformer**(com.ibm.websphere.objectgrid.plugins.ObjectTransformer t) Sets the ObjectTransformer object for use by this BackingMap and/or Loader.
void	**setOptimisticCallback**(com.ibm.websphere.objectgrid.plugins.OptimisticCallback checker) Sets the OptimisticCallback.
void	**setPreloadMode**(boolean async) Sets the preload mode if a Loader is set for this BackingMap.
void	**setReadOnly**(boolean readOnlyEnabled) Sets the map type of this BackingMap.
void	**setTimeBasedDBUpdateConfig**(com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig dbUpdateConfig) Set the time-based database update configuration object.
void	**setTimeToLive**(int seconds) Sets "time to live" of each map entry in seconds.
void	**setTtlEvictorType**(TTLType type) Sets how expiration time of a BackingMap entry is computed.
void	**setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data output format for all data access APIs that return cache values.
void	**setWriteBehind**(String writeBehindParam) Enable write-behind updates for this map.

Methods inherited from interface com.ibm.websphere.objectgrid.ClientReplicableMap
disableClientReplication, enableClientReplication, getReplicationMode

Field Detail

DEFAULT_LOCK_TIMEOUT

`static final int` **`DEFAULT_LOCK_TIMEOUT`**

Default lock timeout used if `setLockTimeout(int)` is not invoked.

See Also:
Constant Field Values

DEFAULT_NUMBER_OF_BUCKETS

`static final int` **`DEFAULT_NUMBER_OF_BUCKETS`**

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*
Default number of lock buckets used if `setNumberOfBuckets(int)` is not invoked.

See Also:
Constant Field Values

DEFAULT_NUMBER_OF_LOCK_BUCKETS

`static final int` **`DEFAULT_NUMBER_OF_LOCK_BUCKETS`**

Default number of lock buckets used if `setNumberOfLockBuckets(int)` is not invoked.

See Also:
Constant Field Values

EVICTIONTRIGGER_MEMORY_USAGE_THRESHOLD

`static final String` **`EVICTIONTRIGGER_MEMORY_USAGE_THRESHOLD`**

The eviction trigger string constant to enable memory based eviction using memory usage threshold provided by the `java.lang.management.MemoryPoolMXBean`.

Since:
WAS XD 6.1.0.3
See Also:
`setEvictionTriggers(String)`, Constant Field Values

LOCAL

`static final int` **`LOCAL`**

Constant used to indicate this map is not a distributed map.

Since:
WAS XD 6.1
See Also:
`getMapType()`, Constant Field Values

SERVER

`static final int` **`SERVER`**

Constant used to indicate this map is a server map.

Since:
WAS XD 6.1
See Also:
`getMapType()`, Constant Field Values

CLIENT

`static final int` **`CLIENT`**

Constant used to indicate this map is a client to a server map

Since:
WAS XD 6.1
See Also:
`getMapType()`, Constant Field Values

Method Detail

getName

`String` **`getName`**`()`

Gets the name of the `BackingMap`.

Returns:
value specified when `BackingMap` was created.

---

setEvictor

`void` **`setEvictor`**`(Evictor e)`

Associates an `Evictor` with this `BackingMap`.

An `Evictor` aids with cleaning up the cache based on whatever algorithm is desired (LRU, LFU, etc). Passing `null` to this method removes a previously set `Evictor` object from an earlier invocation of this method.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

An Evictor that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous evictor which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

An `Evictor` may also implement the `BackingMapPlugin` interface in order to receive enhanced BackingMap plug-in lifecycle method calls. The plug-in is then also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:
`e` - Evictor instance
Throws:
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:
Evictor, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getEvictor

Evictor **getEvictor**()

Gets the Evictor being used by this BackingMap.

Returns:
the argument that was passed to the setEvictor(Evictor) method of this interface or null if setEvictor was not previously called for this BackingMap object.
See Also:
Evictor, setEvictor(Evictor)

---

setObjectTransformer

void **setObjectTransformer**(com.ibm.websphere.objectgrid.plugins.ObjectTransformer t)

Sets the ObjectTransformer object for use by this BackingMap and/or Loader.

An ObjectTransformer aids with the "serialization" of non-Serializable objects. It allows a custom copy function to be installed for more efficient object copy operations.

Note, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

An ObjectTransformer that implements the BackingMapLifecycleListener is automatically added as if the addMapEventListener(EventListener) method was called. Any previous transformer which implements BackingMapLifecycleListener is removed as if the removeMapEventListener(EventListener) method was called.

Parameters:
t - ObjectTransformer instance
Throws:
IllegalArgumentException - if the passed in ObjectTransformer is null
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.
See Also:
ObjectTransformer, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getObjectTransformer

com.ibm.websphere.objectgrid.plugins.ObjectTransformer **getObjectTransformer**()

Gets the ObjectTransformer object being used by this BackingMap and/or Loader.

Returns:
the argument that was passed to the setObjectTransformer(ObjectTransformer) method of this interface or the default ObjectTransformer object if the setObjectTransformer method was not previously called for this object.
See Also:
ObjectTransformer, setObjectTransformer(ObjectTransformer)

---

setOptimisticCallback

`void` **`setOptimisticCallback`**`(com.ibm.websphere.objectgrid.plugins.OptimisticCallback checker)`

Sets the `OptimisticCallback`.

The OptimisticCallback will be used to check the versions of cache entries during the commit phase. If no OptimisticCallback was previously set, a default OptimisticCallback will be used. For Entities, the default OptimisticCallback will use a version field that was specified in the entity metadata. For POJO objects or Entities that do not have a version field specified, the default OptimisticCallback uses the entire object as the version value. In order for it to work for POJO objects, the application's value object needs to have a useful equals(Object) method. If your application does not require versioning, but is using Optimistic locking, the NoVersioningOptimistCallback should be used.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

An OptimisticCallback that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous optimistic callback which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

Parameters:

`checker` - OptimisticCallback instance

Throws:

`IllegalArgumentException` - if the passed in `OptimisticCallback` is `null`

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`OptimisticCallback`, `NoVersioningOptimisticCallback`, `LockStrategy.OPTIMISTIC`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getOptimisticCallback

`com.ibm.websphere.objectgrid.plugins.OptimisticCallback` **`getOptimisticCallback`**`()`

Gets the `OptimisticCallback` being used by this `BackingMap` and/or `Loader` or `null` if the LockStrategy is not optimistic.

If no OptimisticCallback was previously set, a default OptimisticCallback will be used. For Entities, the default OptimisticCallback will use a version field that was specified in the entity metadata. For POJO objects or Entities that do not have a version field specified, the default OptimisticCallback uses the entire object as the version value. In order for it to work for POJO objects, the application's value object needs to have a useful equals(Object) method. If your application does not require versioning, but is using Optimistic locking, the NoVersioningOptimistCallback should be used.

Returns:

the argument that was passed to the `setOptimisticCallback(OptimisticCallback)` method of this interface or the default `OptimisticCallback` object if the `setOptimisticCallback` method was not previously called for this object. If Optimistic locking is not being used, this method will return null after ObjectGrid.initialize() has been invoked.

See Also:

`NoVersioningOptimisticCallback`, `OptimisticCallback`, `LockStrategy.OPTIMISTIC`, `setOptimisticCallback(OptimisticCallback)`

---

setLoader

`void` **`setLoader`**`(com.ibm.websphere.objectgrid.plugins.Loader loader)`

Associates a `Loader` with this `BackingMap`.

Only one Loader can be associated with a given `BackingMap`. Passing `null` to this method removes a previously set `Loader` object from an earlier invocation of this method and indicates that this `BackingMap` is not associated with a `Loader`.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

A loader that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous loader which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

A `Loader` may also implement the `BackingMapPlugin` interface in order to receive enhanced BackingMap plug-in lifecycle method calls. The plug-in is then also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:

`loader` - Loader instance

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`Loader`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getLoader

`com.ibm.websphere.objectgrid.plugins.Loader` **`getLoader`**`()`

Gets the Loader being used by this `BackingMap`.

Returns:

the argument that was passed to the `setLoader(Loader)` method of this interface or `null` if `setLoader` was not previously called for this object.

See Also:

`Loader`, `setLoader(Loader)`

---

setPreloadMode

`void` **`setPreloadMode`**`(boolean async)`

Sets the preload mode if a `Loader` is set for this `BackingMap`.

If the parameter is true then the `Loader.preloadMap(Session, BackingMap)` is invoked asynchronously; otherwise it blocks the execution when loading data so the cache is unavailable until preload completes. Preloading occurs during `ObjectGrid` initialization.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`async` - If this is true then the cache is loaded asynchronously otherwise it blocks and the cache is unavailable until preload completes.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()`

method is called.

See Also:

`Loader.preloadMap(Session, BackingMap)`

---

getPreLoadMode

`boolean` **`getPreLoadMode`**`()`

Returns whether this `BackingMap` will be asynchronously preloaded or not if a `Loader` is set.

If true is returned then the `Loader.preloadMap(Session, BackingMap)` method is invoked asynchronously; otherwise it blocks the execution when loading data so the cache is unavailable until preload completes. Preloading occurs during `ObjectGrid` initialization.

Returns:

the argument that was passed to the `setPreloadMode(boolean)` method of this interface or `false` if setPreloadeMode was not previously called for this object.

See Also:

`Loader.preloadMap(Session, BackingMap)`, `setPreloadMode(boolean)`

---

addMapIndexPlugin

```
void addMapIndexPlugin(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index)
                       throws com.ibm.websphere.objectgrid.IndexAlreadyDefinedException
```

Adds an `MapIndexPlugin` to this Map. This method assumes the index implementation was constructed with the name of the attribute to index. The name of the index is specified when the index is constructed.

Note, to avoid an `IllegalStateException`, this method must be called prior to `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

A MapIndexPlugin that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous index which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

Parameters:

`index` - The index implementation.

Throws:

`IndexAlreadyDefinedException` - if this index already exists.

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`MapIndexPlugin`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getMapIndexPlugins

`List` **`getMapIndexPlugins`**`()`

Returns the current list of `MapIndexPlugin` objects for this BackingMap.

Returns:

The current list of `MapIndexPlugin`s for this BackingMap. The list is empty if the `addMapIndexPlugin(MapIndexPlugin)` or `setMapIndexPlugins(List)` method was not previously called for this BackingMap.

See Also:

addMapIndexPlugin(MapIndexPlugin), setMapIndexPlugins(List)

---

setMapIndexPlugins

void **setMapIndexPlugins**(List indexList)

Sets the list of MapIndexPlugin objects for this BackingMap. If the BackingMap already has a List of MapIndexPlugin objects, that list is replaced by the List passed as an argument to the current invocation of this method.

Note, to avoid an IllegalStateException, this method must be called prior to ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:

indexList - A non-null reference to a List of MapIndexPlugin objects.

Throws:

IllegalArgumentException - is thrown if indexList is null or the indexList contains either a null reference or an object that is not an instance of MapIndexPlugin.

See Also:

MapIndexPlugin, ObjectGrid.initialize(), ObjectGrid.getSession()

---

setCopyMode

void **setCopyMode**(CopyMode mode,
Class valueInterface)

Sets the CopyMode.

The CopyMode determines whether a get operation of an entry in the BackingMap returns the actual value, a copy of the value, or a proxy for the value. In the case of a proxy, the copy of the value does not occur unless a set method of the application provided value interface is invoked. It also determines that when a transaction is committed, whether a copy of the value object of an entry that was marked as dirty by the transaction is put into the BackingMap at commit time. The CopyMode does not specify if the object is copied when being read or written to a Loader. It is the responsibility of the implementor of a Loader to make copies as appropriate. The default CopyMode is CopyMode.COPY_ON_READ_AND_COMMIT.

Note, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:

mode - must be one of the final static variables defined in CopyMode. See CopyMode class for an explanation of each mode and how the valueInterface is used for CopyMode.COPY_ON_WRITE .

valueInterface - the value interface Class object. Specify null in version 7.1 and later.

Throws:

IllegalArgumentException - if mode is CopyMode.COPY_ON_WRITE and valueInterface parameter is null and CGLIB isn't in the classpath.

IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

See Also:

CopyMode, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getCopyMode

CopyMode **getCopyMode**()

Gets the `CopyMode` being used by this `BackingMap`.

Returns:
the argument that was passed to the `setCopyMode(CopyMode, Class)` method of this interface or the default `CopyMode` object if `setCopyMode` was not previously called for this object.

See Also:
CopyMode, setCopyMode(CopyMode, Class)

---

setLockStrategy

void **setLockStrategy**(LockStrategy lockStrategy)

Sets the `LockStrategy`.

The locking strategy represented by the `LockStrategy` object determines if the internal ObjectGrid lock manager is used whenever a map entry is accessed by a transaction. The default strategy is `LockStrategy.OPTIMISTIC`.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:
`lockStrategy` - must be one of the final static variables defined in `LockStrategy`. See `LockStrategy` class for an explanation of each locking strategy.

Throws:
IllegalStateException - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:
LockStrategy, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getLockStrategy

LockStrategy **getLockStrategy**()

Gets the `LockStrategy` object being used by this `BackingMap`.

Returns:
the argument that was passed to the `setLockStrategy(LockStrategy)` method of this interface or the default `LockStrategy` object if `setLockStrategy` was not previously called for this object.

See Also:
LockStrategy, setLockStrategy(LockStrategy)

---

setMapEventListeners

@Deprecated
void **setMapEventListeners**(List eventListenerList)

Deprecated. *This method is deprecated in version 7.1.1. Use the addMapEventListener(EventListener) or removeMapEventListener(EventListener) methods. Plugins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*

Sets the list of `EventListener` objects.

If this BackingMap already has a List of EventListenerS, that list is replaced by the List passed as an argument to the current invocation of this method. This method can be called before and after the ObjectGrid.initialize() method.

Parameters:

eventListenerList - A non-null reference to a List of EventListener objects that are instances of BackingMapLifecycleListener or MapEventListener

Throws:

IllegalArgumentException - is thrown if eventListenerList is null, the eventListenerList contains either a null reference or an object that is not an instance of BackingMapLifecycleListener or MapEventListener

See Also:

EventListener, MapEventListener, BackingMapLifecycleListener, addMapEventListener(EventListener), removeMapEventListener(EventListener)

---

getMapEventListeners

List **getMapEventListeners**()

Gets the current list of EventListenerS.

Returns:

the current list of EventListener objects for this BackingMap.

See Also:

EventListener, MapEventListener, BackingMapLifecycleListener

---

addMapEventListener

void **addMapEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener eventListener)

Adds an EventListener to this BackingMap.

Note, this method is allowed to be invoked before and after the ObjectGrid.initialize() method. Backing map plug-ins (Loader, Evictor, MapIndexPlugin, ObjectTransformer, OptimisticCallback) that implement the ObjectGridLifecycleListener are automatically added as listeners when added to the BackingMap.

Parameters:

eventListener - A non-null reference to a EventListener to add to the list. The listener must be an instance of BackingMapLifecycleListener or MapEventListener

Throws:

IllegalArgumentException - if eventListener is null or not an instance of BackingMapLifecycleListener or MapEventListener

See Also:

EventListener, MapEventListener, BackingMapLifecycleListener

---

addMapEventListener

void **addMapEventListener**(MapEventListener eventListener)

Deprecated. *This method is deprecated in version 7.1.1, use the addMapEventListener(EventListener) method.*

Provided for compatibility with old releases, use the addMapEventListener(EventListener) method.

Parameters:

eventListener - A non-null reference to a EventListener to add to the list. The listener must be an instance of BackingMapLifecycleListener or MapEventListener

removeMapEventListener

`void` **`removeMapEventListener`**`(com.ibm.websphere.objectgrid.plugins.EventListener eventListener)`

Removes an `EventListener` from this `BackingMap`.

Note, this method is allowed to be invoked before and after the `ObjectGrid.initialize()` method. Backing map plug-ins (`Loader`, `Evictor`, `MapIndexPlugin`, `ObjectTransformer`, `OptimisticCallback`) that implement the `ObjectGridLifecycleListener` are automatically removed as listeners when removed from the `ObjectGrid`.

Parameters:

`eventListener` - A non-null reference to an event listener that was previously added by invoking either the `addMapEventListener(EventListener)` or `setMapEventListeners(List)` method of this interface.

Throws:

`IllegalArgumentException` - if eventListener is null or not an instance of `BackingMapLifecycleListener` or `MapEventListener`

See Also:

`EventListener`, `MapEventListener`, `BackingMapLifecycleListener`, `addMapEventListener(EventListener)`

---

removeMapEventListener

`void` **`removeMapEventListener`**`(MapEventListener eventListener)`

Provided for compatibility with old releases, use the `removeMapEventListener(EventListener)` method.

Parameters:

`eventListener` - A non-null reference to an event listener that was previously added by invoking either the `addMapEventListener(EventListener)` or `setMapEventListeners(List)` method of this interface.

---

getPartitionId

`int` **`getPartitionId`**`()`

Gets the partition identifier being used by this `BackingMap`.

Returns:

The 0-based index for the partition represented by this `BackingMap` instance. If there is only a single partition defined for this `BackingMap` object, a 0 will be returned (default).

Since:

WAS XD 6.0.1

---

setReadOnly

`void` **`setReadOnly`**`(boolean readOnlyEnabled)`

Sets the map type of this `BackingMap`.

A map can be a read only map or a read/write map. Passing true as the parameter value will make this map a read only map; passing false as the parameter value will make this map a read/write map.

Note, to avoid an `IllegalStateException`, this method must be called prior to the

`ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`readOnlyEnabled` - If set to true, this `BackingMap` will be a read only map. If false, the map will be a read/write map.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

---

getReadOnly

`boolean` **`getReadOnly`**`()`

Retrieves the map type.

Returns:

the argument that was passed to `setReadOnly(boolean)` method of this interface. True is returned if this a read only map. A return value of false implies that this is a read/write map. If `setReadOnly` was never called, the default return value is false.

See Also:

`setReadOnly(boolean)`

---

getObjectGrid

`ObjectGrid` **`getObjectGrid`**`()`

Gets the `ObjectGrid` that owns this `BackingMap`.

Returns:

the ObjectGrid instance that owns this BackingMap.

See Also:

`ObjectGrid`

---

setNumberOfBuckets

`void` **`setNumberOfBuckets`**`(int numBuckets)`

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*

Sets the number of buckets used by this `BackingMap`.

The `BackingMap` implementation uses a hash map for its implementation. If there are a lot of entries in the `BackingMap` then more buckets means better performance because the risk of collisions is lower as the number of buckets grows. More buckets also means more concurrency. If number of buckets is 0, no entries will be stored in the map, but the appropriate ObjectGrid and BackingMap plug-ins will still be called.

Once the `ObjectGrid` is initialized this parameter cannot be changed. Therefore, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`numBuckets` - The number of buckets to use.

Throws:

IllegalArgumentException - if numBuckets is less than 0.
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

See Also:

ObjectGrid.initialize(), ObjectGrid.getSession()

---

getNumberOfBuckets

int **getNumberOfBuckets**()

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the isNearCacheEnabled() flag to disable the near cache in the ObjectGrid configuration XML file.*

Gets the number of buckets defined for this BackingMap.

Returns:

the same value passed to the setNumberOfBuckets(int) method or DEFAULT_NUMBER_OF_BUCKETS if setNumberOfBuckets was never called.

See Also:

setNumberOfBuckets(int), DEFAULT_NUMBER_OF_BUCKETS

---

setNumberOfLockBuckets

void **setNumberOfLockBuckets**(int numBuckets)

Sets the number of lock buckets used by the lock manager for this BackingMap.

When LockStrategy.OPTIMISTIC or LockStrategy.PESSIMISTIC is used for this BackingMap, a lock manager is created for the BackingMap. The lock manager uses a hash map to keep track of entries that are locked by 1 or more transactions. If there are a lot of entries in the hash map, then more lock buckets means better performance as the risk of collisions is lower as the number of buckets grows. More lock buckets also means more concurrency. When the lock strategy is LockStrategy.NONE, no lock manager is used by this BackingMap. In this case, a call to this method does nothing.

Once the ObjectGrid is initialized, the number of lock buckets cannot be changed. Therefore, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:

numBuckets - The number of lock buckets to use.

Throws:

IllegalArgumentException - if numBuckets is less than 1.
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

See Also:

LockStrategy, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getNumberOfLockBuckets

int **getNumberOfLockBuckets**()

Gets the number of lock buckets defined for the hash map used by lock manager for this backing map.

Returns:

the same value passed to the `setNumberOfLockBuckets(int)` method or `DEFAULT_NUMBER_OF_LOCK_BUCKETS` if `setNumberOfLockBuckets` was never called.

See Also:

`setNumberOfLockBuckets(int)`, `DEFAULT_NUMBER_OF_LOCK_BUCKETS`

---

setLockTimeout

`void` **`setLockTimeout`**`(int seconds)`

Sets the lock timeout used by the lock manager for this `BackingMap`.

When `LockStrategy.OPTIMISTIC` or `LockStrategy.PESSIMISTIC` is used for this `BackingMap`, a lock manager is created for the `BackingMap`. To prevent deadlocks from occurring, the lock manager has a default timeout value for waiting for a lock to be granted. If this timeout limit is exceeded, a `LockTimeoutException` is thrown. The default value of `DEFAULT_LOCK_TIMEOUT` should be sufficient for most applications, but on a heavily loaded system, a timeout may occur when no deadlock exists. In that case, this method can be used to increase the lock timeout value from the default to whatever is desired to prevent false timeout exceptions from occurring. When the lock strategy is `LockStrategy.NONE`, no lock manager is used by this `BackingMap`. In this case, a call to this method does nothing. A lock timeout value of zero indicates to not wait for the lock if it is not immediately available.

Once the lock manager is initialized, the lock timeout value cannot be changed. Therefore, to avoid an `IllegalStateException`, this method must be called prior to `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application. When an entry is fetched the lock timeout can be changed for a given transaction using `ObjectMap.setLockTimeout(int)`

Parameters:

`seconds` - is the lock timeout value to use in seconds.

Throws:

`IllegalArgumentException` - if seconds is less than 0.

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`DEFAULT_LOCK_TIMEOUT`, `LockStrategy`, `LockTimeoutException`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`, `ObjectMap.setLockTimeout(int)`

---

getLockTimeout

`int` **`getLockTimeout`**`()`

Gets the lock timeout value used by the lock manager for this `BackingMap`.

Returns:

the same value passed to the `setLockTimeout(int)` method or `DEFAULT_LOCK_TIMEOUT` if `setLockTimeout` was never called.

See Also:

`DEFAULT_LOCK_TIMEOUT`, `setLockTimeout(int)`

---

setNullValuesSupported

`void` **`setNullValuesSupported`**`(boolean nullValuesSupported)`

Sets whether this `BackingMap` supports `null` values.

If `null` values are supported, users need to be careful when a `get` operation returns a `null`

reference. It could be due to the fact that the key is not found in the `BackingMap`, or that the value in the `BackingMap` is `null`. To determine if a key was not found, or the value is `null`, the `containsKey` method can be used.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`nullValuesSupported` - If set to true, `null` values are supported; otherwise `null` values are not supported.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`ObjectGrid.initialize()`, `ObjectGrid.getSession()`, `ObjectMap.containsKey(Object)`

---

getNullValuesSupported

`boolean **getNullValuesSupported**()`

Gets whether this `BackingMap` supports `null` values or not.

Returns:

the same value passed to the `setNullValuesSupported(boolean)` method or the default value of true if `setNullValuesSupported` was never called.

See Also:

`setNullValuesSupported(boolean)`

---

setCopyKey

`void **setCopyKey**(boolean copy)`

Sets whether or not the key needs to be copied when a map entry is created.

Copying the key object allows the application to use the same key object for each ObjectMap operation. The application changes the key object state prior to each `ObjectMap` operation so that it can work with different entries using the same key object. If a separate key object is used for each entry, then there is no reason to copy the key object. This attribute allows an application to make the tradeoff of copying key object versus using more memory as a result of separate key object used by the application for each entry. If this method is not called, then the default of `false` is used (e.g. the key is NOT copied).

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`copy` - If `true` is specified, then this `BackingMap` uses the `ObjectTransformer.copyKey(Object)` method to copy the key object when necessary.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`ObjectGrid.initialize()`, `ObjectGrid.getSession()`, `ObjectTransformer.copyKey(Object)`

---

getCopyKey

`boolean` **`getCopyKey`**`()`

Gets whether keys are copied for this `BackingMap`.

Returns:
the same value passed to the `setCopyKey(boolean)` method or the default value of false if `setCopyKey` was never called.

See Also:
`setCopyKey(boolean)`

---

setTimeToLive

`void` **`setTimeToLive`**`(int seconds)`

Sets "time to live" of each map entry in seconds.

If this method is not called, the lifetime of an entry is forever (or until the application explicitly removes or invalidates the entry, or a user defined `Evictor` evicts the entry). Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:
`seconds` - the number of seconds a map entry is allowed to live in map before being evicted.

Throws:
`IllegalArgumentException` - if seconds is less than 0.
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:
`setTtlEvictorType(TTLType)`, `ObjectMap.setTimeToLive(int)`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getTimeToLive

`int` **`getTimeToLive`**`()`

Gets the number of seconds for an entry to live.

This value returned is in seconds and 0 indicates forever.

Returns:
the same value passed to the `setTimeToLive(int)` method or 0 if `setLockTimeout` was never called.

See Also:
`setTimeToLive(int)`

---

setTtlEvictorType

`void` **`setTtlEvictorType`**`(TTLType type)`

Sets how expiration time of a `BackingMap` entry is computed.

If this method is not called, `TTLType.NONE` is used to indicate the map entry has no expiration time (e.g. is allowed to live until explicitly removed or invalidated by the application, or evicted by a user defined `Evictor`).

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`type` - must be one of the public constants declared in the `TTLType` class.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`TTLType`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getTtlEvictorType

`TTLType` **`getTtlEvictorType`**`()`

Gets how expiration time of a `BackingMap` entry is computed.

Returns:

the `TTLType` that was passed to the `setTtlEvictorType(TTLType)` or `TTLType.NONE` if `setTtlEvictorType` was never called.

See Also:

`setTtlEvictorType(TTLType)`, `TTLType`

---

createDynamicIndex

```
void createDynamicIndex(String name,
                        boolean isRangeIndex,
                        String attributeName,
                        com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicInd
exCallback)
                        throws com.ibm.websphere.objectgrid.IndexAlreadyDefinedException,
                            IllegalArgumentException
```

Creates a dynamic index on the `BackingMap`.

Required Client Permission: `ServerMapPermission.REPLICATE`

Parameters:

`name` - the name of the index. The name can not be null or a zero length string.
`isRangeIndex` - Indicate whether to create a `MapRangeIndex` or a `MapIndex`. If set to true, the index will be a type of `MapRangeIndex`.
`attributeName` - The name of the attribute to be indexed. The attributeName can not be null or a zero length string.
`dynamicIndexCallback` - The callback that will invoke upon dynamic index events. The dynamicIndexCallback is optional and can be null.

Throws:

`IllegalArgumentException` - if name or attributeName is null or a zero length string.
`IndexAlreadyDefinedException` - if a `MapIndexPlugin` with the specified name already exists.

Since:

WAS XD 6.0.1

See Also:

`MapIndex`, `MapIndexPlugin`, `MapRangeIndex`, `ObjectMap.getIndex(String)`

---

createDynamicIndex

```
void createDynamicIndex(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index,
```

```
                    com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicInd
exCallback)
                    throws com.ibm.websphere.objectgrid.IndexAlreadyDefinedException,
                           IllegalArgumentException
```

Creates a dynamic index on the `BackingMap`.

Required Client Permission: `ServerMapPermission.DYNAMIC_INDEX`

A MapIndexPlugin that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous index which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

Parameters:

`index` - The index implementation. The index can not be `null`.
`dynamicIndexCallback` - The callback that will invoke upon dynamic index events. The dynamicIndexCallback is optional and can be null.

Throws:

`IllegalArgumentException` - if index is `null` or `index.getName()` returns `null` or a zero length string.
`IndexAlreadyDefinedException` - if a `MapIndexPlugin` with the specified name already exists.

Since:

WAS XD 6.0.1

See Also:

`MapIndexPlugin`, `ObjectMap.getIndex(String)`

---

removeDynamicIndex

```
void removeDynamicIndex(String name)
                    throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                           IllegalArgumentException
```

Removes a dynamic index on the `BackingMap`.

Required Client Permission: `ServerMapPermission.DYNAMIC_INDEX`

Parameters:

`name` - the name of the index. The name can not be `null`.

Throws:

`IllegalArgumentException` - if name is `null`.
`IndexUndefinedException` - if a `MapIndexPlugin` with the specified name does not exists.

Since:

WAS XD 6.0.1

See Also:

`createDynamicIndex(MapIndexPlugin, DynamicIndexCallback)`, `createDynamicIndex(String, boolean, String, DynamicIndexCallback)`

---

getPartitionManager

```
PartitionManager getPartitionManager()
```

Allows access to the `PartitionManager` that is defined for this `BackingMap`. This access may be useful for `Loaders during Loader.preloadMap(Session, BackingMap) processing (to properly partition the data to be loaded).`

Returns:

PartitionManager associated with this BackingMap.

Since:

WAS XD 6.0.1

See Also:

PartitionManager, Loader.preloadMap(Session, BackingMap)

---

getEntityMetadata

com.ibm.websphere.projector.md.EntityMetadata **getEntityMetadata**()

Retreive the metadata for the entity associated with this backing map.

Returns:
the EntityMetadata if an entity is associated with this backing map or null if there is no entity associated with this backing map.

Since:
WAS XD 6.1

---

setWriteBehind

void **setWriteBehind**(String writeBehindParam)

Enable write-behind updates for this map.

If a map is configured with write-behind loader update, the updates (could be insert type, remove type, or update type) to the backend are not instantly updated to the back end by calling the Loader.batchUpdate(TxID, LogSequence) method. Instead, they are queued in a write-behind queue map and updated to the back end periodically.

A write-behind update is pushed to the backend periodically within a different transaction from the one the update is made to ObjectGrid. When the write-behind update to the backend fails, for example, due to data integrity problem, it is too late to roll back the original ObjectGrid transaction. ObjectGrid will invalidate the entry and create an entry in a failed database update map. The name of this failed database update map is WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX+baseMapName. The key of the entry in this map is an auto-increment Integer, and the value is a LogElement. The logElement can be used to compensate the failure.

Depending on your grid use case and your back end loader configuration, your back end loader or the back end database might benefit from having upsert operations instead of insert and update operations in the LogElements that it receives for a transaction. Use the ConvertToUpsert=true configuration option on the writeBehindParam to have the write behind loader convert insert and update LogElement operations to upsert LogElement operations when they are passed to the back end loader. Not all back end loaders may support the upsert operation, be certain that the back end loader supports upsert operations before using the ConvertToUpsert=true clause in the writeBehindParam. The default value is ConvertToUpsert=false.

Note, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:
writeBehindParam - a write-behind parameter consisting of a maximum update time and/or a maximum key update count. The format of the write-behind parameter is "T[time];C[count][;ConvertToUpsert=true]", for example, "T100;C2000". "T100;C2000" means the loader will write to the back end when there are 2000 pending keys to be updated or when 100 seconds have passed since the last update. The default update time is 300 seconds and the default update key count is 1000. You can configure the update time only, the update key count only, or an empty string. The default value(s) will then be used in either of the above three cases. The default value is null to disable write-behind updates.

Throws:
IllegalArgumentException - if the write behind parameters are unknown or improperly formatted.
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

Since:

WAS XD 6.1.0.3

See Also:

`WriteBehindLoaderConstants`

---

getWriteBehind

`String` **`getWriteBehind`**`()`

Get the write-behind parameter. A write-behind parameter consists of a maximum update time and/or a maximum key update count. The format of the write-behind parameter is "T[time];C[count][;ConvertToUpsert=true]".

Returns:

the write-behind parameter. If the write-behind parameter is not set, `null` will be returned.

Since:

WAS XD 6.1.0.3

See Also:

`setWriteBehind(String)`

---

setTimeBasedDBUpdateConfig

`void` **`setTimeBasedDBUpdateConfig`**`(com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig dbUpdateConfig)`

Set the time-based database update configuration object.

When a time-based database update configuration object is set, a thread will be started automatically to update or invalidate the ObjectGrid maps with the latest updates (inserts and updates) from the database.

For a local ObjectGrid map, the thread will be launched in the same JVM. For a distributed ObjectGrid map in an ObjectGrid container, the thread will be automatically launched in partition 0. No database update thread will be started in a client side near cache.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`dbUpdateConfig` - the time-based database update configuration object or `null`.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

Since:

WAS XD 6.1.0.3

---

getTimeBasedDBUpdateConfig

`com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig` **`getTimeBasedDBUpdateConfig`**`()`

Get the time-based database update configuration object.

Returns:

the time-based database update configuration object or `null` if not set.

Since:

WAS XD 6.1.0.3

See Also:
setTimeBasedDBUpdateConfig(TimeBasedDBUpdateConfig)

---

getMapType

`int` **`getMapType`**`()`

Returns the type of BackingMap.

The return value is equivalent to one of the constants declared on this interface, `LOCAL`, `SERVER`, or `CLIENT`.

Returns:
the map type
Since:
WAS XD 6.1

---

getEvictionTriggers

`String` **`getEvictionTriggers`**`()`

Returns the types of additional eviction triggers.

The available eviction trigger strings are a described in the String constants in this interface that begin with the name: `EVICTIONTRIGGER`.

Returns:
a semicolon separated list of eviction triggers
Since:
WAS XD 6.1.0.3

---

setEvictionTriggers

`void` **`setEvictionTriggers`**`(String evictionTriggers)`

Sets the types of additional eviction triggers, all evictors for the backing map will use the provided set of triggers.

The available eviction trigger strings are a described in the String constants in this interface that begin with the name: `EVICTIONTRIGGER`.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:
`evictionTriggers` - a semicolon separated list of eviction triggers
Throws:
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.
`IllegalArgumentException` - if the eviction triggers are unknown or improperly formatted.
Since:
WAS XD 6.1.0.3

---

getMapSetName

`String` **`getMapSetName`**()

Retrieves the name of the MapSet that this BackingMap is currently associated with. A "null" return value indicates it is currently not associated with a MapSet. This method will only return a non null value for a client or server map.

Returns:
name of associated MapSet
Since:
7.1
See Also:
`getMapType()`

---

getSerializerAccessor

`com.ibm.websphere.objectgrid.plugins.io.SerializerAccessor` **`getSerializerAccessor`**()

Retrieve the SerializerAccessor for this map.

Returns:
the SerializerAccessor
Since:
7.1.1

---

getState

`com.ibm.websphere.objectgrid.plugins.BackingMapLifecycleListener.State` **`getState`**()

Retrieve the current life cycle state of this map.

Returns:
the current state.
Since:
7.1.1

---

isNearCacheInvalidationEnabled

`boolean` **`isNearCacheInvalidationEnabled`**()

If true, clients with local caches are automatically invalidated when the data grid map is updated.

Returns:
true if client near cache invalidation is enabled.
Since:
8.6, XC10 2.5

---

isNearCacheLastAccessTTLSyncEnabled

`boolean` **`isNearCacheLastAccessTTLSyncEnabled`**()

If true, clients automatically send time-to-live access information to the remote data grid when accessed and the `TTLType.LAST_ACCESS_TIME` TTL evictor type is configured.

Returns:
true if last-access time-to-live information is sent to the remote data grid.
Since:
8.6, XC10 2.5

isNearCacheEnabled

`boolean` **`isNearCacheEnabled`**`()`

If true, the client near cache is enabled for supported configurations. The client near cache can only be enabled when using optimistic locking or when locking is disabled.

Returns:
true if the client near cache is enabled.

---

getKeyOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getKeyOutputFormat`**`()`

Retrieves the data format for all data access APIs that return cache keys.

This value does not reflect the data output format that plug-ins will see. See the `PluginOutputFormat` annotation for details on how to influence the data object format that plug-ins receive.

Returns:
the data output format.
Since:
8.6, XC10 2.5

---

setKeyOutputFormat

`void` **`setKeyOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`

Sets the data output format for all data access APIs that return cache keys.

When set to `OutputFormat.UNDEFINED`, the key output format defaults to `OutputFormat.RAW` when using a custom `KeyDataSerializer` plug-in. The key output format is `OutputFormat.NATIVE` in all other cases.

Parameters:
`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default.
Throws:
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.
Since:
8.6, XC10 2.5

---

getValueOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getValueOutputFormat`**`()`

Retrieves the data output format for all data access APIs that return cache values.

This value does not reflect the data output format that plug-ins will see. See the `PluginOutputFormat` annotation for details on how to influence the data object format that plug-ins receive.

Returns:
the data output format or `OutputFormat.UNDEFINED` if the default should be used.
Since:
8.6, XC10 2.5

setValueOutputFormat

`void` **`setValueOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`

Sets the data output format for all data access APIs that return cache values.

When set to `OutputFormat.UNDEFINED`, the value output format defaults to `OutputFormat.RAW` when using a custom `ValueDataSerializer` plug-in or when the `CopyMode.COPY_TO_BYTES_RAW` CopyMode set.

The value output format is `OutputFormat.NATIVE` in all other cases.

Parameters:

`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

Since:

8.6, XC10 2.5

com.ibm.websphere.objectgrid

Class AvailabilityState

java.lang.Object
  com.ibm.websphere.objectgrid.AvailabilityState

All Implemented Interfaces:

Serializable

```
public final class AvailabilityState
extends Object
implements Serializable
```

Each shard in a distributed ObjectGrid has an availability state associated with it. This state refers to the shard's ability to process incoming requests.

Since:

WAS XD 6.1.0.3, XC10

See Also:

Serialized Form

Field Summary

static AvailabilityState	**OFFLINE** An AvailabilityState.OFFLINE indicates that a shard is offline and unable to process requests.
static AvailabilityState	**ONLINE**

[bilityState](#)	An AvailabilityState.ONLINE indicates that a shard is online.
static [AvailabilityState](#)	**PRELOAD** An AvailabilityState.PRELOAD indicates that a shard is in the preload state.
static [AvailabilityState](#)	**QUIESCE** An AvailabilityState.QUIESCE indicates that a shard is in quiesce.
static [AvailabilityState](#)	**UNKNOWN** An AvailabilityState.UNKNOWN indicates that the availability state of the shard could not be determined.

State	

Method Summary

int	**getId**() Returns the internal identifier for this state.
String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

OFFLINE

```
public static final AvailabilityState OFFLINE
```

An AvailabilityState.OFFLINE indicates that a shard is offline and unable to process requests.

PRELOAD

```
public static final AvailabilityState PRELOAD
```

An AvailabilityState.PRELOAD indicates that a shard is in the preload state. When in the preload state, a shard will reject all requests that are not initiated from a client that is preloading data into the ObjectGrid.

ONLINE

```
public static final AvailabilityState ONLINE
```

An AvailabilityState.ONLINE indicates that a shard is online. A shard that is online is available for processing requests.

QUIESCE

```
public static final AvailabilityState QUIESCE
```

An AvailabilityState.QUIESCE indicates that a shard is in quiesce. Quiesce is a transitional state. Shards that are in the quiesce state are on their way to being offline. A shard in the quiesce state will allow all pending transactions to complete before moving to the AvailabilityState.OFFLINE, assuming that all pending transactions complete within 30 seconds after entering the quiesce state.

UNKNOWN

`public static final AvailabilityState UNKNOWN`

An AvailabilityState.UNKNOWN indicates that the availability state of the shard could not be determined.

Method Detail

getId

`public int getId()`

Returns the internal identifier for this state.

Returns:
the internal id.

Since:
7.1.1

---

toString

`public String toString()`

Overrides:
`toString` in class `Object`

Package com.ibm.websphere.xs.ra

These are the eXtreme Scale Resource Adapter APIs that allows integration with a Java EE application.

See:

Description

Interface Summary

ObjectGridJ2CConnectionMBean	Allows administration clients control the lifecycle of the eXtreme Scale resource adapter's connection.

Class Summary

XSConnection	An XSConnection represents an application level connection handle to an eXtreme Scale ObjectGrid.
XSConnectionFactory	The XSConnectionFactory creates connections to eXtreme Scale ObjectGrids.
XSConnectionSpec	The ConnectionSpec for retrieving XSConnections from an XSConnectionFactory
XSManagedConnectionFactory	The XSManagedConnectionFactory create managed connections to the eXtreme Scale ObjectGrids.
XSResourceAdapter	The eXtreme Scale resource adapter

Package com.ibm.websphere.xs.ra Description

These are the eXtreme Scale Resource Adapter APIs that allows integration with a Java EE application.

Introduction

The eXtreme Scale Resource Adapter provides client connection management and local transaction support, allowing Java EE applications to look-up eXtreme Scale client connections and demarcate transactions using the `LocalTransaction` interface or the `Session` interface.

When used with WebSphere Application Server with last participant support enabled, the eXtreme Scale transaction can be enlisted in a global transaction as the last, single-phase participant.

Programming Tutorial

The following sections show snippets on the usage of the ConnectionFactory. The resource

adapter is is JCA 1.5 compliant. To use the resource adapter the following steps must be followed:

1. Install the resource adapter
2. Configure a J2C ConnectionFactory
3. Configure a javax.resource.cci.ConnectionFactory resource reference in the application.
4. Look-up the `XSConnectionFactory`
5. Choose one of the following transaction options:

   Use auto-commit, local transactions:

   1. Retrieve a `XSConnection`
   2. Retrieve and use the `Session` to interact with the data grid.
   3. Close the connection.

   Use an ObjectGrid Session to demarcate a local transaction:

   1. Retrieve a `XSConnection`
   2. Retrieve the `Session`
   3. Use the Session.begin() method to start the transaction.
   4. Use the Session to interact with the data grid.
   5. Use the Session.commit() or rollback()) methods to end the transaction.
   6. Close the connection.

   Use a javax.resource.cci.LocalTransction to demarcate a local transaction:

   1. Retrieve a `XSConnection`
   2. Retrieve the javax.resource.cci.LocalTransaction using the XSConnection.getLocalTransaction() method.
   3. Use the LocalTransaction.begin() method to start the transaction.
   4. Retrieve and use the `Session` to interact with the data grid.
   5. Use the LocalTransaction.commit() or rollback() methods to end the transaction.
   6. Close the connection.

   Enlist the connection in a global transaction:

   1. Lookup the UserTransction.
   2. Begin the global transaction
   3. Retrieve a `XSConnection`
   4. Retrieve and use the `Session`
   5. Close the connection.
   6. Commit or rollback the global transaction.

Installing the resource adapter

The eXtreme Scale resource adapter is included in the wxsra.rar resource adapter archive with the eXtreme Scale product. See the WebSphere eXtreme Scale version 8.5 information center (or later) for details on how to install and configure the resource adapter.

Configuring the J2C ConnectionFactory

The eXtreme Scale resource adapter allows configuring one or J2C ManagedConnectionFactory instances. Each ManagedConnectionFactory is managed by the application server and represents a connection to a single catalog service domain. the ManagedConnectionFactory can include the name of the data grid, or the data grid can be provided when the connection is retrieved by the application.

The ManagedConnectionFactory is configured in the application server configuration, bound to a global JNDI name, and provides the following configuration properties:

ConnectionNa	

me	(Optional) The name of the eXtreme Scale client connection.
CatalogServiceEndpoints	The catalog service domain end points in the form: <host>:<port>,<host><port>. Required if catalog service domain is not set.
CatalogServiceDomain	The catalog service domain name. Required if catalog service end points are not set.
ObjectGridName	(Optional) The data grid name. If not specified, the client must provide the data grid name when retrieving the connection resource.
ObjectGridURL	(Optional) The URL of the client data grid, override XML file.
ObjectGridResource	(Optional) The resource path of the client data grid, override XML file.
ClientPropertiesURL	(Optional) The URL of the client properties file.
ClientPropertiesResource	(Optional) The resource path of the client properties file.

Obtaining a ConnectionFactory instance.

The Java EE application can use resource injection to inject a ConnectionFactory resource into the application, or it can be looked-up using a resource reference. The Java EE application must first configure resource reference for a javax.resource.cci.ConnectionFactory.

For example:

```
InitialContext ctx = new InitialContext();
XSConnectionFactory cf = (XSConnectionFactory) ctx.lookup("java:comp/env/wxsconnection");
```

Retrieving a Connection

After the `XSConnectionFactory` has been looked-up or injected into the application, use one of the getConnection() methods to retrieve a client connection to the data grid. The connection will automatically be established when the first connection is retrieved and will be maintained until the resource adapter is stopped or the connection is reset using the ObjectGridJ2CConnection management bean.

For example:

```
XSConnection con = cf.getConnection("MyGrid");
```

Obtaining an ObjectGrid Session instance.

The XSConnection provides a getSession() method that gives the application direct access to the ObjectGrid Session. The Session is used to interact with the data grid and is valid for the life of the XSConnection. The XSConnection is a handle to a connection and becomes invalid after the application context completes per the Java EE specification.

An eXtreme Scale local transaction can be driven by the Session, javax.resource.cci.LocalTransaction or a global transaction. The transaction methods cannot be mixed.

Closing a connection

After the application has finished using the connection, the connection must be closed. The Java EE container typically will also close the connection automatically at the appropriate times. When the connection is closed the Session and any other objects retrieved directlly or indirectly from the connection become invalid.

---

com.ibm.websphere.xs.ra

Class XSManagedConnectionFactory

java.lang.Object
  com.ibm.websphere.xs.ra.XSManagedConnectionFactory

All Implemented Interfaces:

Serializable, ManagedConnectionFactory, ResourceAdapterAssociation

---

```
public final class XSManagedConnectionFactory
extends Object
implements ManagedConnectionFactory, ResourceAdapterAssociation
```

The XSManagedConnectionFactory create managed connections to the eXtreme Scale ObjectGrids.

Since:

8.5, XC10

See Also:

Serialized Form

Constructor Summary

XSManagedConnectionFactory()

Method Summary

Object	**createConnectionFactory**() Creates a non-managed XSConnectionFactory instance.
Object	**createConnectionFactory**(ConnectionManager conMgr) Creates a managed XSConnectionFactory instance.
ManagedConnection	**createManagedConnection**(Subject subject, ConnectionRequestInfo conReq) Creates a new physical connection to the underlying ObjectGrid

[ction](#)	
void	**destroy**() Destroy this MCF and release any other resources.
boolean	**equals**(Object obj)
String	**getCatalogServiceDomain**() Get the eXtreme Scale specific property: CatalogDomain
String	**getCatalogServiceEndpoints**() Get the eXtreme Scale specific property: CatalogServiceEndpoints
String	**getClientPropertiesResource**() Get the eXtreme Scale specific property: ClientPropertiesResource
String	**getClientPropertiesURL**() Get the eXtreme Scale specific property: ClientPropertiesURL
String	**getConnectionName**() Get the eXtreme Scale specific property: ConnectionName
PrintWriter	**getLogWriter**() Get the log writer for this XSManagedConnectionFactory instance.
String	**getObjectGridName**() Get the eXtreme Scale specific property: ObjectGridName

String	**getObjectGridResource**() Get the eXtreme Scale specific property: ObjectGridResource
String	**getObjectGridURL**() Get the eXtreme Scale specific property: ObjectGridURL
ResourceAdapter	**getResourceAdapter**() Retrieve the associated eXtreme Scale resource adapter instance.
int	**hashCode**()
boolean	**isLocalGrid**() Retrieve the eXtreme Scale specific property: LocalGrid
ManagedConnection	**matchManagedConnections**(Set connectionSet, Subject subject, ConnectionRequestInfo cxRequestInfo) Returns a matched connection from the candidate set of connections.
void	**setCatalogServiceDomain**(String catalogServiceDomain) Set the eXtreme Scale specific property: CatalogDomain
void	**setCatalogServiceEndpoints**(String catalogServiceEndpoints) Set the eXtreme Scale specific property: CatalogServiceEndpoints
vo	**setClientPropertiesResource**(String clientPropsResource) Get the eXtreme Scale specific property: ClientPropertiesResource

void	**setClientPropertiesURL**(String clientPropsURL) Set the eXtreme Scale specific property: ClientPropertiesURL
void	**setConnectionName**(String connName) Set the eXtreme Scale specific property: ConnectionName
void	**setLocalGrid**(boolean useLocal) Set the eXtreme Scale specific property: LocalGrid
void	**setLogWriter**(PrintWriter out) Set the log writer for this ManagedConnectionFactory instance
void	**setObjectGridName**(String objectGridName) Set the eXtreme Scale specific property: ObjectGrid Name
void	**setObjectGridResource**(String objectGridResource) Set the eXtreme Scale specific property: ObjectGridResource
void	**setObjectGridURL**(String objectGridURL) Set the eXtreme Scale specific property: ObjectGridURL
void	**setResourceAdapter**(ResourceAdapter ra) Associate this object with an eXtreme Scale resource adapter.
String	**toString**()

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

XSManagedConnectionFactory

public **XSManagedConnectionFactory**()

Method Detail

getResourceAdapter

public ResourceAdapter **getResourceAdapter**()

Retrieve the associated eXtreme Scale resource adapter instance.

Specified by:
`getResourceAdapter` in interface `ResourceAdapterAssociation`

Returns:
The associated eXtreme Scale resource adapter

See Also:
`ResourceAdapterAssociation.getResourceAdapter()`

---

setResourceAdapter

```
public void setResourceAdapter(ResourceAdapter ra)
                        throws ResourceException
```

Associate this object with an eXtreme Scale resource adapter. Note, this method must be called exactly once. That is, the association must not change during the lifetime of this object.

Specified by:
`setResourceAdapter` in interface `ResourceAdapterAssociation`

Parameters:
`ra` - the eXtreme Scale resource adapter

Throws:
`ResourceException`
`IllegalStateException`

See Also:
`ResourceAdapterAssociation.setResourceAdapter(ResourceAdapter)`

---

createConnectionFactory

```
public Object createConnectionFactory()
                                throws ResourceException
```

Creates a non-managed XSConnectionFactory instance. The XSConnectionFactory instance gets initialized with a default XSConnectionManager instance provided by the eXtreme Scale resource adapter.

Specified by:
`createConnectionFactory` in interface `ManagedConnectionFactory`

Returns:
the new XSConnectionFactory instance

Throws:
`ResourceException`

See Also:
`ManagedConnectionFactory.createConnectionFactory()`

---

createConnectionFactory

```
public Object createConnectionFactory(ConnectionManager conMgr)
                                throws ResourceException
```

Creates a managed XSConnectionFactory instance. The XSConnectionFactory instance gets initialized with the passed ConnectionManager. In the managed scenario, ConnectionManager is provided by the application server.

Specified by:
`createConnectionFactory` in interface `ManagedConnectionFactory`

Parameters:

`conMgr` - - Connection Manager to be associated with the XSConnectionFactory.

Returns:

the new XSConnectionFactory instance

Throws:

`ResourceException` - ResourceAdapterInternalException

See Also:

`ManagedConnectionFactory.createConnectionFactory(ConnectionManager)`

---

createManagedConnection

```
public ManagedConnection createManagedConnection(Subject subject,
                                                 ConnectionRequestInfo conReq)
                                          throws ResourceException
```

Creates a new physical connection to the underlying ObjectGrid

Specified by:

`createManagedConnection` in interface `ManagedConnectionFactory`

Parameters:

`subject` - - Caller's security information. WebSphere eXtreme Scale client uses the credentials specified in a CredentialGenerator as part of the ConnectionSpec or the client properties.

`conReq` - - XS specific connection properties

Returns:

the new XSManagedConnection instance

Throws:

`ResourceException`

See Also:

`ManagedConnectionFactory.createManagedConnection(Subject, ConnectionRequestInfo)`

---

getLogWriter

```
public PrintWriter getLogWriter()
                           throws ResourceException
```

Get the log writer for this XSManagedConnectionFactory instance.

Specified by:

`getLogWriter` in interface `ManagedConnectionFactory`

Returns:

PrintWriter

Throws:

`ResourceException`

See Also:

`ManagedConnectionFactory.getLogWriter()`

---

matchManagedConnections

```
public ManagedConnection matchManagedConnections(Set connectionSet,
                                                 Subject subject,
                                                 ConnectionRequestInfo cxRequestInfo)
                                          throws ResourceException
```

Returns a matched connection from the candidate set of connections.

Specified by:

`matchManagedConnections` in interface `ManagedConnectionFactory`

Parameters:

`connectionSet` - - candidate connection set

`subject` - - caller's security information

`cxRequestInfo` - - XS specific connection properties

Returns:

XSManagedConnection instance if acceptable match found otherwise null

Throws:

ResourceException

See Also:

ManagedConnectionFactory.matchManagedConnections(Set, Subject, ConnectionRequestInfo)

---

setLogWriter

```
public void setLogWriter(PrintWriter out)
                  throws ResourceException
```

Set the log writer for this ManagedConnectionFactory instance

Specified by:

setLogWriter in interface ManagedConnectionFactory

Parameters:

`out` - - PrintWriter - an out stream for error logging and tracing

Throws:

ResourceException

See Also:

ManagedConnectionFactory.setLogWriter(PrintWriter)

---

destroy

```
public void destroy()
```

Destroy this MCF and release any other resources.

---

getConnectionName

```
public String getConnectionName()
```

Get the eXtreme Scale specific property: ConnectionName

Returns:

The name of the eXtreme Scale client connection

---

setConnectionName

```
public void setConnectionName(String connName)
```

Set the eXtreme Scale specific property: ConnectionName

Parameters:

`connName` - - The name of the eXtreme Scale client connection

---

getObjectGridName

```
public String getObjectGridName()
```

Get the eXtreme Scale specific property: ObjectGridName

Returns:
The data grid name

---

setObjectGridName

`public void` **`setObjectGridName`**`(`[String](#) `objectGridName)`

Set the eXtreme Scale specific property: ObjectGrid Name

Parameters:
`objectGridName` - - The data grid name

---

getCatalogServiceEndpoints

`public` [String](#) **`getCatalogServiceEndpoints`**`()`

Get the eXtreme Scale specific property: CatalogServiceEndpoints

Returns:
The catalog service domain end points

---

setCatalogServiceEndpoints

`public void` **`setCatalogServiceEndpoints`**`(`[String](#) `catalogServiceEndpoints)`

Set the eXtreme Scale specific property: CatalogServiceEndpoints

Parameters:
`catalogServiceEndpoints` - - The catalog service domain end points

---

getCatalogServiceDomain

`public` [String](#) **`getCatalogServiceDomain`**`()`

Get the eXtreme Scale specific property: CatalogDomain

Returns:
The catalog service domain name defined in WebSphere Application Server

---

setCatalogServiceDomain

`public void` **`setCatalogServiceDomain`**`(`[String](#) `catalogServiceDomain)`

Set the eXtreme Scale specific property: CatalogDomain

Parameters:
`catalogServiceDomain` - - The catalog service domain name defined in WebSphere Application Server

---

getObjectGridURL

`public` [String](#) **`getObjectGridURL`**`()`

Get the eXtreme Scale specific property: ObjectGridURL

Returns:
The URL of the client data grid override XML file

setObjectGridURL

public void **setObjectGridURL**(String objectGridURL)

Set the eXtreme Scale specific property: ObjectGridURL

Parameters:

objectGridURL - The URL of the client data grid override XML file

---

getObjectGridResource

public String **getObjectGridResource**()

Get the eXtreme Scale specific property: ObjectGridResource

Returns:

The resource path of the client data grid override XML file

---

setObjectGridResource

public void **setObjectGridResource**(String objectGridResource)

Set the eXtreme Scale specific property: ObjectGridResource

Parameters:

objectGridResource - - The resource path of the client data grid override XML file

---

getClientPropertiesURL

public String **getClientPropertiesURL**()

Get the eXtreme Scale specific property: ClientPropertiesURL

Returns:

The URL of the client properties file

---

setClientPropertiesURL

public void **setClientPropertiesURL**(String clientPropsURL)

Set the eXtreme Scale specific property: ClientPropertiesURL

Parameters:

clientPropsURL - - The URL of the client properties file

---

getClientPropertiesResource

public String **getClientPropertiesResource**()

Get the eXtreme Scale specific property: ClientPropertiesResource

Returns:

The resource path of the client properties file

---

setClientPropertiesResource

public void **setClientPropertiesResource**(String clientPropsResource)

Get the eXtreme Scale specific property: ClientPropertiesResource

Parameters:

`clientPropsResource` - - The resource path of the client properties file

---

setLocalGrid

public void **setLocalGrid**(boolean useLocal)

Set the eXtreme Scale specific property: LocalGrid

When set to true, the application uses the XSConnectionSpec.setLocalObjectGrid(com.ibm.websphere.objectgrid.ObjectGrid) to set the ObjectGrid instance to a local, in-memory grid instance or shard instance. If set to false (the default), the connection will be configured as a client connection to a remote data grid.

Parameters:

`useLocal` - set to true, to disable normal client ObjectGrid connection management.

Since:

8.6, XC10 2.5

---

isLocalGrid

public boolean **isLocalGrid**()

Retrieve the eXtreme Scale specific property: LocalGrid

Returns:

answers true if the connection is to be used with a local, im-memory ObjectGrid instance. Answers false if the connection represents a normal client ObjectGrid connection.

Since:

8.6, XC10 2.5

---

hashCode

public int **hashCode**()

Specified by:

`hashCode` in interface `ManagedConnectionFactory`

Overrides:

`hashCode` in class `Object`

See Also:

ManagedConnectionFactory.hashCode()

---

equals

public boolean **equals**(Object obj)

Specified by:

`equals` in interface `ManagedConnectionFactory`

Overrides:

`equals` in class `Object`

See Also:

ManagedConnectionFactory.equals(Object)

---

toString

public String **toString**()

Overrides:

toString in class Object

---

com.ibm.websphere.xs.ra

Class XSConnectionSpec

java.lang.Object
  com.ibm.websphere.xs.ra.XSConnectionSpec

All Implemented Interfaces:
Cloneable, ConnectionSpec

```
public final class XSConnectionSpec
extends Object
implements ConnectionSpec, Cloneable
```

The ConnectionSpec for retrieving XSConnections from an XSConnectionFactory

Since:
8.5, XC10

See Also:
XSConnectionFactory

Constructor Summary

XSConnectionSpec() Creates a default XSCOnnectionSpec instance.

Method Summary

Object	**clone**()
boolean	**equals**(Object obj)
CredentialGenerator	**getCredentialGenerator**() Returns the CredentialGenerator object for this client security for the connection.

[erator](#)	
[Object](#)[Grid](#)	**getLocalObjectGrid**() Retrieve the local object grid instance used for the connection, if set.
[String](#)	**getObjectGridName**() Returns the object grid name for the connection.
int	**hashCode**()
boolean	**isBeginNoWriteThrough**() Returns flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.
boolean	**isMultiPartitionSupportEnabled**() Answers true if the [Session](#) will have the [Session.getTxCommitProtocol()](#) set to [Session.TxCommitProtocol.TWOPHASE](#).
void	**setBeginNoWriteThrough**(boolean beginNoWriteThrough) Set the boolean flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.
void	**setCredentialGenerator**([CredentialGenerator](#) credGen) Sets the `CredentialGenerator` object for the client security for the connection.
void	**setLocalObjectGrid**([ObjectGrid](#) instance) Sets a local ObjectGrid instance to be used for connections.
void	**setMultiPartitionSupportEnabled**(boolean mptEnabled) Set to true to retrieve a connection with a [Session](#) that has the [Session.getTxCommitProtocol()](#) set to [Session.TxCommitProtocol.TWOPHASE](#).
void	**setObjectGridName**([String](#) objectGridName) Sets the object grid name to be used for the connection.
[St](#)	

[r i n g](#)	**toString**()

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

XSConnectionSpec

public **XSConnectionSpec**()

Creates a default XSCOnnectionSpec instance.

Method Detail

isBeginNoWriteThrough

public boolean **isBeginNoWriteThrough**()

Returns flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.

Returns:

flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.

---

setBeginNoWriteThrough

public void **setBeginNoWriteThrough**(boolean beginNoWriteThrough)

Set the boolean flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.

Parameters:

`beginNoWriteThrough` -

See Also:

Session.begin(), Session.beginNoWriteThrough()

---

getObjectGridName

public String **getObjectGridName**()

Returns the object grid name for the connection.

Returns:

object grid name

---

setObjectGridName

public void **setObjectGridName**(String objectGridName)

Sets the object grid name to be used for the connection.

Parameters:

`objectGridName` - object grid name to be used for the connection

---

getLocalObjectGrid

`public ObjectGrid **getLocalObjectGrid**()`

Retrieve the local object grid instance used for the connection, if set.

Returns:

null if the connection retrieves its ObjectGrid instance normally from the physical connection, or non-null if `setLocalObjectGrid(ObjectGrid)` was specified.

Since:

8.6, XC10 2.5

---

setLocalObjectGrid

`public void **setLocalObjectGrid**(ObjectGrid instance)`

Sets a local ObjectGrid instance to be used for connections. The local ObjectGrid instance overrides the standard behavior and allows a physical connection to be used with a specific local ObjectGrid instance, such as a shard.

Parameters:

`instance` - the local ObjectGrid instance.

Since:

8.6, XC10 2.5

---

getCredentialGenerator

`public CredentialGenerator **getCredentialGenerator**()`

Returns the `CredentialGenerator` object for this client security for the connection.

Returns:

the argument that was passed to the `setCredGen(CredentialGenerator)` method of this object or `null` if `setCredGen` was not previously called for this object.

---

setCredentialGenerator

`public void **setCredentialGenerator**(CredentialGenerator credGen)`

Sets the `CredentialGenerator` object for the client security for the connection.

Parameters:

`credGen` - a CredentialGenerator object

See Also:

`CredentialGenerator`

---

setMultiPartitionSupportEnabled

`public void **setMultiPartitionSupportEnabled**(boolean mptEnabled)`

Set to true to retrieve a connection with a `Session` that has the `Session.getTxCommitProtocol()` set to `Session.TxCommitProtocol.TWOPHASE`.

Parameters:

mptEnabled - set to true if the Session should be configured to write to multiple partitions.

Since:

8.6, XC10 2.5

---

isMultiPartitionSupportEnabled

`public boolean` **`isMultiPartitionSupportEnabled`**`()`

Answers true if the `Session` will have the `Session.getTxCommitProtocol()` set to `Session.TxCommitProtocol.TWOPHASE`.

Returns:

true if the Session is capable of writing to multiple partitions.

Since:

8.6, XC10 2.5

---

clone

`public Object` **`clone`**`()`

Overrides:

`clone` in class `Object`

---

hashCode

`public int` **`hashCode`**`()`

Overrides:

`hashCode` in class `Object`

---

equals

`public boolean` **`equals`**`(Object obj)`

Overrides:

`equals` in class `Object`

---

toString

`public String` **`toString`**`()`

Overrides:

`toString` in class `Object`

---

com.ibm.websphere.xs.ra

Class XSResourceAdapter

java.lang.Object
  com.ibm.websphere.xs.ra.XSResourceAdapter

All Implemented Interfaces:

ResourceAdapter

```
public final class XSResourceAdapter
extends Object
implements ResourceAdapter
```

The eXtreme Scale resource adapter

Since:

8.5, XC10

Constructor Summary

XSResourceAdapter()

Method Summary

void	**endpointActivation**(MessageEndpointFactory msgEndpointFactory, ActivationSpec actSpec) This method does nothing as endpoint activation is not supported
void	**endpointDeactivation**(MessageEndpointFactory msgEndpointFactory, ActivationSpec spec) This method does nothing as endpoint deactivation is not supported
XAResource[]	**getXAResources**(ActivationSpec[] spec) This method returns null as the XA protocol is not supported
void	**start**(BootstrapContext ctx) This method initializes the eXtreme Scale resource adapter instance
void	**stop**()

id	This method performs the shutdown of the eXtreme Scale resource adapter instance
String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XSResourceAdapter

```
public XSResourceAdapter()
```

Method Detail

endpointActivation

```
public void endpointActivation(MessageEndpointFactory msgEndpointFactory,
                               ActivationSpec actSpec)
                        throws ResourceException
```

This method does nothing as endpoint activation is not supported

Specified by:
endpointActivation in interface ResourceAdapter

Throws:
ResourceException

See Also:
ResourceAdapter.endpointActivation(MessageEndpointFactory,ActivationSpec)

---

endpointDeactivation

```
public void endpointDeactivation(MessageEndpointFactory msgEndpointFactory,
                                 ActivationSpec spec)
```

This method does nothing as endpoint deactivation is not supported

Specified by:
endpointDeactivation in interface ResourceAdapter

Throws:
ResourceException

See Also:
ResourceAdapter.endpointDeactivation(MessageEndpointFactory,ActivationSpec)

---

getXAResources

```
public XAResource[] getXAResources(ActivationSpec[] spec)
                            throws ResourceException
```

This method returns null as the XA protocol is not supported

Specified by:
getXAResources in interface ResourceAdapter

Throws:
ResourceException

See Also:
ResourceAdapter.getXAResources(ActivationSpec[])

---

start

```
public void start(BootstrapContext ctx)
          throws ResourceAdapterInternalException
```

This method initializes the eXtreme Scale resource adapter instance

Specified by:
start in interface ResourceAdapter

Throws:
ResourceAdapterInternalException

See Also:
ResourceAdapter.start(BootstrapContext)

---

stop

```
public void stop()
```

This method performs the shutdown of the eXtreme Scale resource adapter instance

Specified by:
stop in interface ResourceAdapter

See Also:
ResourceAdapter.stop()

---

toString

```
public String toString()
```

Overrides:
toString in class Object

---

com.ibm.websphere.xs.ra

Class XSConnection

java.lang.Object
  com.ibm.websphere.xs.ra.XSConnection

All Implemented Interfaces:
Connection

```
public final class XSConnection
extends Object
implements Connection
```

An XSConnection represents an application level connection handle to an eXtreme Scale ObjectGrid.

Since:
8.5, XC10

Constructor Summary

XSConnection()

Method Summary

void	**close**() Initiates close of the connection handle at the application level.
Interaction	**createInteraction**() Creates an Interaction associated with this Connection.
LocalTransaction	**getLocalTransaction**() Returns an LocalTransaction instance that enables a component to demarcate local transactions on the Connection.

[tion](#)	
[ConnectionMetaData](#)	**[getMetaData](#)**() Gets the information on the underlying eXtreme Scale instance represented through an active connection.
[ResultSetInfo](#)	**[getResultSetInfo](#)**() Gets the information on the ResultSet functionality supported by a connected eXtreme Scale instance.
[Session](#)	**[getSession](#)**() Retrieve the ObjectGrid Session associated with this connection.
[String](#)	**[toString](#)**()

Methods inherited from class java.lang.[Object](#)
[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Constructor Detail

XSConnection

public **XSConnection**()

Method Detail

getSession

```
public Session getSession()
                  throws ResourceException
```

Retrieve the ObjectGrid Session associated with this connection.

Returns:
The ObjectGrid Session
Throws:
ResourceException

---

close

```
public void close()
                  throws ResourceException
```

Initiates close of the connection handle at the application level.

Specified by:
close in interface Connection

Throws:
ResourceException
See Also:
Connection.close()

---

createInteraction

```
public Interaction createInteraction()
                           throws ResourceException
```

Creates an Interaction associated with this Connection.

Specified by:
createInteraction in interface Connection

Returns:
Interaction instance
Throws:
ResourceException
NotSupportedException
See Also:
Connection.createInteraction()

---

getLocalTransaction

```
public LocalTransaction getLocalTransaction()
                                    throws ResourceException
```

Returns an LocalTransaction instance that enables a component to demarcate local transactions on the Connection.

Specified by:
getLocalTransaction in interface Connection

Returns:
LocalTransaction instance
Throws:
ResourceException
See Also:
Connection.getLocalTransaction()

---

getMetaData

```
public ConnectionMetaData getMetaData()
                            throws ResourceException
```

Gets the information on the underlying eXtreme Scale instance represented through an active connection.

Specified by:
getMetaData in interface Connection

Returns:
WXS ConnectionMetaData

Throws:
ResourceException

See Also:
Connection.getMetaData()

---

getResultSetInfo

```
public ResultSetInfo getResultSetInfo()
                              throws ResourceException
```

Gets the information on the ResultSet functionality supported by a connected eXtreme Scale instance.

Specified by:
getResultSetInfo in interface Connection

Returns:
ResultSetInfo instance

Throws:
ResourceException

See Also:
Connection.getResultSetInfo()

---

toString

```
public String toString()
```

Overrides:
toString in class Object

---

com.ibm.websphere.xs.ra

Class XSConnectionFactory

java.lang.Object
  com.ibm.websphere.xs.ra.XSConnectionFactory

All Implemented Interfaces:
Serializable, Referenceable, ConnectionFactory

```
public final class XSConnectionFactory
extends Object
implements ConnectionFactory
```

The XSConnectionFactory creates connections to eXtreme Scale ObjectGrids.

Usage example:

```
InitialContext ctx = new InitialContext();
XSConnectionFactory cf = (XSConnectionFactory) ctx.lookup("java:comp/env/wxsconnection");
XSConnection con = cf.getConnection("MyGrid");
Session ogSession = con.getSession();
...
con.close();
```

For additional examples, see the package documentation.

Since:
8.5, XC10

See Also:
Serialized Form

Constructor Summary

XSConnectionFactory()

Method Summary

String	**getCatalogServiceDomain**() Get the eXtreme Scale specific property: CatalogDomain
String	**getCatalogServiceEndpoints**() Get the eXtreme Scale specific property: CatalogServiceEndpoints

g	
String	**getClientPropertiesResource**() Get the eXtreme Scale specific property: ClientPropertiesResource
String	**getClientPropertiesURL**() Get the eXtreme Scale specific property: ClientPropertiesURL
Connection	**getConnection**() Retrieve an XSConnection for the ObjectGrid configured on the ConnectionFactory.
Connection	**getConnection**(ConnectionSpec spec) Retrieve an XSConnection for the ObjectGrid using the specified XSConnectionSpec properties.
Connection	**getConnection**(String gridName) Retrieve an XSConnection for the specified ObjectGrid name.
String	**getConnectionName**() Get the eXtreme Scale specific property: ConnectionName
ResourceAdapterMet	**getMetaData**() Retrieve the metadata information regarding the eXtreme Scale resource adapter

[aData](#)	
[String](#)	**getObjectGridName**() Get the eXtreme Scale specific property: ObjectGridName
[String](#)	**getObjectGridResource**() Get the eXtreme Scale specific property: ObjectGridResource
[String](#)	**getObjectGridURL**() Get the eXtreme Scale specific property: ObjectGridURL
[RecordFactory](#)	**getRecordFactory**() Retrieve the RecordFactory instance
[Reference](#)	**getReference**() Gets the Reference instance
boolean	**isLocalGrid**() Get the eXtreme Scale specific property: LocalGrid
void	**setReference**([Reference](#) ref) Sets the Reference instance
[String](#)	**toString**()

Methods inherited from class java.lang.[Object](#)

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XSConnectionFactory

public **XSConnectionFactory**()

Method Detail

setReference

public void **setReference**(Reference ref)

Sets the Reference instance

Parameters:
ref - - Reference instance
See Also:
Referenceable.setReference(Reference)

---

getReference

public Reference **getReference**()
throws NamingException

Gets the Reference instance

Specified by:
getReference in interface Referenceable

Returns:
Reference instance
Throws:
NamingException
See Also:
Referenceable.getReference()

---

getConnection

public Connection **getConnection**()
throws ResourceException

Retrieve an XSConnection for the ObjectGrid configured on the ConnectionFactory.

Specified by:
getConnection in interface ConnectionFactory

Returns:
the connection to the eXtreme Scale ObjectGrid
Throws:
ResourceException
See Also:
ConnectionFactory.getConnection()

---

getConnection

```
public Connection getConnection(ConnectionSpec spec)
                         throws ResourceException
```

Retrieve an XSConnection for the ObjectGrid using the specified XSConnectionSpec properties.

Specified by:

getConnection in interface ConnectionFactory

Parameters:

spec - the XSConnectionSpec properties used to retrieve the correct connection.

Returns:

the connection to the eXtreme Scale ObjectGrid

Throws:

ResourceException

See Also:

ConnectionFactory.getConnection(ConnectionSpec), XSConnectionSpec

---

getConnection

```
public Connection getConnection(String gridName)
                         throws ResourceException
```

Retrieve an XSConnection for the specified ObjectGrid name.

Parameters:

gridName - - The ObjectGrid name

Returns:

the connection to the eXtreme Scale ObjectGrid

Throws:

ResourceException

---

getMetaData

```
public ResourceAdapterMetaData getMetaData()
                                  throws ResourceException
```

Retrieve the metadata information regarding the eXtreme Scale resource adapter

Specified by:

getMetaData in interface ConnectionFactory

Returns:

The eXtreme Scale resource adapter metadata

Throws:

ResourceException

See Also:

ConnectionFactory.getMetaData()

---

getRecordFactory

```
public RecordFactory getRecordFactory()
                               throws ResourceException
```

Retrieve the RecordFactory instance

Specified by:

getRecordFactory in interface ConnectionFactory

Throws:

NotSupportedException
ResourceException

See Also:

ConnectionFactory.getRecordFactory()

---

toString

public String **toString**()

Overrides:

toString in class Object

---

getCatalogServiceDomain

public String **getCatalogServiceDomain**()

Get the eXtreme Scale specific property: CatalogDomain

Returns:

The catalog service domain name defined in WebSphere Application Server

---

getCatalogServiceEndpoints

public String **getCatalogServiceEndpoints**()

Get the eXtreme Scale specific property: CatalogServiceEndpoints

Returns:

The catalog service domain end points

---

getClientPropertiesResource

public String **getClientPropertiesResource**()

Get the eXtreme Scale specific property: ClientPropertiesResource

Returns:

The resource path of the client properties file

---

getClientPropertiesURL

public String **getClientPropertiesURL**()

Get the eXtreme Scale specific property: ClientPropertiesURL

Returns:

The URL of the client properties file

---

getConnectionName

public String **getConnectionName**()

Get the eXtreme Scale specific property: ConnectionName

Returns:

The name of the eXtreme Scale client connection

---

getObjectGridName

public String **getObjectGridName**()

Get the eXtreme Scale specific property: ObjectGridName

Returns:
The data grid name

---

getObjectGridResource

public String **getObjectGridResource**()

Get the eXtreme Scale specific property: ObjectGridResource

Returns:
The resource path of the client data grid override XML file

---

getObjectGridURL

public String **getObjectGridURL**()

Get the eXtreme Scale specific property: ObjectGridURL

Returns:
The URL of the client data grid override XML file

---

isLocalGrid

public boolean **isLocalGrid**()

Get the eXtreme Scale specific property: LocalGrid

Returns:
Is this managed connection factory used only for access to a local ObjectGrid instance?

---

com.ibm.websphere.xs.ra

Interface ObjectGridJ2CConnectionMBean

public interface **ObjectGridJ2CConnectionMBean**

Allows administration clients control the lifecycle of the eXtreme Scale resource adapter's connection.

The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGridJ2CConnection,objectGridName=<objectgrid>,domain=<dom
ain name>,connectionName=<connection name>
```

Note: Additional properties may be included.

Since:

8.5, XC10

Field Summary

static String	**CONNECTIONNAME_DEFAULT** The default connection name prefix when not specified.
static String	**CONNECTIONSTATUS_CONNECTED** The connection status indicating that a connection has been established.
static String	**CONNECTIONSTATUS_DISCONNECTED** The connection status indicating that client is no longer connected.
s	

static String	**MBEAN_TYPE** The MBean type

Method Summary

String	**getCatalogServiceEndpoints**() The catalog service endpoints of the catalog service.
String	**getConnectionName**() The name of the connection as specified on the J2C ConnectionFactory, or "DEFAULT" if not specified.
String	**getConnectionStatus**() The status of the connection.
String	**getDomainName**() The domain name of the catalog service domain as reported by the catalog service.
String	**getObjectGridName**() The name of the ObjectGrid that is connected.
void	**resetObjectGridConnection**() Reset the ObjectGrid connection.

Field Detail

MBEAN_TYPE

```
static final String MBEAN_TYPE
```

The MBean type

See Also:

Constant Field Values

CONNECTIONSTATUS_CONNECTED

static final String **CONNECTIONSTATUS_CONNECTED**

The connection status indicating that a connection has been established.

See Also:
Constant Field Values

CONNECTIONSTATUS_DISCONNECTED

static final String **CONNECTIONSTATUS_DISCONNECTED**

The connection status indicating that client is no longer connected.

See Also:
Constant Field Values

CONNECTIONNAME_DEFAULT

static final String **CONNECTIONNAME_DEFAULT**

The default connection name prefix when not specified.

See Also:
Constant Field Values

Method Detail

resetObjectGridConnection

void **resetObjectGridConnection**()

Reset the ObjectGrid connection.

This destroys the client connection to the ObjectGrid, including any local cache that may be created.

Subsequent uses of the ManagedConnectionFactory will result in a new ObjectGrid connection.

---

getObjectGridName

String **getObjectGridName**()

The name of the ObjectGrid that is connected.

Returns:
the ObjectGrid name

---

getConnectionName

String **getConnectionName**()

The name of the connection as specified on the J2C ConnectionFactory, or "DEFAULT" if not specified. If there are multiple connections with the same attributes in the same process, the "DEFAULT" name will have an integer appended.

Returns:
the connection name.

---

getDomainName

`String` **`getDomainName`**`()`

The domain name of the catalog service domain as reported by the catalog service.

Returns:
the catalog service domain name.

---

getCatalogServiceEndpoints

`String` **`getCatalogServiceEndpoints`**`()`

The catalog service endpoints of the catalog service.

Returns:
the catalog service endpoints

---

getConnectionStatus

`String` **`getConnectionStatus`**`()`

The status of the connection. Valid states include "CONNECTED" or "DISCONNECTED".

Other states may be returned in the future.

Returns:
one of the CONNECTIONSTATUS constants defined in this interface.

Package com.ibm.websphere.xsa

Class Summary	
RestValue	**Deprecated.** *Use the `RestValue` interface and the `RestValueFactory`.*

Overview | Package | Class | Tree | Serialized | Deprecated | Index | Help

PREV PACKAGE NEXT PACKAGE | FRAMES | NO FRAMES | All Classes

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

com.ibm.websphere.xsa

Class RestValue

java.lang.Object
  com.ibm.websphere.xsa.RestValue

All Implemented Interfaces:
com.ibm.websphere.xs.rest.RestValue, Externalizable, Serializable

---

Deprecated. *Use the `RestValue` interface and the `RestValueFactory`.*

```
public class RestValue
extends Object
implements com.ibm.websphere.xs.rest.RestValue, Externalizable
```

This object is used in the XC10 REST Gateway. It is a data wrapper for data to be accessed via XC10 REST Gateway. With its wrapped data, it will be stored in an object grid's map entry, with a String as the key of the map entry.

Since:
8.6, XC10 1.0.0.4
See Also:
Serialized Form

Field Summary

protected boolean	**compressed** **Deprecated.** Flag to indicate if the value is compressed or not
protected String	**contentType** **Deprecated.** Content type of the contained value

g	
protected byte[]	**value** **Deprecated.** Contained value

Constructor Summary

RestValue() **Deprecated.** Constructor
RestValue(byte[] value, String contentType) **Deprecated.** Constructor

Method Summary

boolean	**equals**(Object obj) **Deprecated.**
String	**getContentType**() **Deprecated.** Return content type of the contained value
byte[]	**getValue**() **Deprecated.** Return contained value
int	**hashCode**() **Deprecated.**
boolean	**isCompressed**() **Deprecated.** Indicate if the contained value is compressed
void	**readExternal**(ObjectInput in) **Deprecated.**
vo	**setCompressed**(boolean compressed)

void	**Deprecated.** Set indicator to indicate if the contained value is compressed
void	**setContentType**(String contentType) **Deprecated.** Set content type of the contained value
void	**setValue**(byte[] value) **Deprecated.** Set contained value
String	**toString**() **Deprecated.**
void	**writeExternal**(ObjectOutput out) **Deprecated.**

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

value

protected byte[] **value**

Deprecated.
Contained value

---

contentType

protected String **contentType**

Deprecated.
Content type of the contained value

---

compressed

protected boolean **compressed**

Deprecated.
Flag to indicate if the value is compressed or not

Constructor Detail

RestValue

public **RestValue**()

Deprecated.

Constructor

Since:
8.6, XC10 1.0.0.4

RestValue

```
public RestValue(byte[] value,
                 String contentType)
```

Deprecated.
Constructor

Parameters:
`value` - A value to be wrapped by RestValue class.
`contentType` - Content type of the value.

Since:
8.6, XC10 1.0.0.4

Method Detail

hashCode

```
public int hashCode()
```

Deprecated.

Overrides:
hashCode in class Object

---

equals

```
public boolean equals(Object obj)
```

Deprecated.

Overrides:
equals in class Object

---

toString

```
public String toString()
```

Deprecated.

Overrides:
toString in class Object

---

getValue

```
public byte[] getValue()
```

Deprecated.
Return contained value

Specified by:
`getValue` in interface `com.ibm.websphere.xs.rest.RestValue`

Returns:
An array of byte that represents the contained value
Since:
8.6, XC10 1.0.0.4

---

setValue

```
public void setValue(byte[] value)
```

Deprecated.
Set contained value

Parameters:
`value` - An array of byte that represents the contained value
Since:
8.6, XC10 1.0.0.4

---

getContentType

```
public String getContentType()
```

Deprecated.
Return content type of the contained value

Specified by:
`getContentType` in interface `com.ibm.websphere.xs.rest.RestValue`

Returns:
A content type
Since:
8.6, XC10 1.0.0.4

---

setContentType

```
public void setContentType(String contentType)
```

Deprecated.
Set content type of the contained value

Parameters:
`contentType` - A content type
Since:
8.6, XC10 1.0.0.4

---

readExternal

```
public void readExternal(ObjectInput in)
                  throws IOException,
                         ClassNotFoundException
```

Deprecated.

Specified by:
`readExternal` in interface `Externalizable`

Throws:
`IOException`
`ClassNotFoundException`

writeExternal

```
public void writeExternal(ObjectOutput out)
                   throws IOException
```

Deprecated.

Specified by:

writeExternal in interface Externalizable

Throws:

IOException

---

isCompressed

```
public boolean isCompressed()
```

Deprecated.
Indicate if the contained value is compressed

Returns:

true if the contained value is compressed

Since:

8.6, XC10 1.0.0.4

---

setCompressed

```
public void setCompressed(boolean compressed)
```

Deprecated.
Set indicator to indicate if the contained value is compressed

Parameters:

`compressed` - A boolean value that indicate if the contained value is compressed

---

IBM WebSphere eXtreme Scale Client for .NET API Specification

Contents

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Namespaces

Namespace	Description
	The IBM.WebSphere.Caching namespace is the parent namespace for the the WebSphere eXtreme Scale C# client data access application programming interfaces. The primary entry point for interacting with the data grid is the GridManagerFactory, which provides access to the IGridManager and IGrid interfaces.
IBM.WebSphere.Caching	The IGridManager interface provides methods to connect and disconnect from a catalog service domain. It also provides access to IGrid instances from the catalog service domain connections. The IGrid interface allows clients to interact with a named data grid using various map interfaces, such as the IGridMapPessimisticTx TKey, TValue and IGridMapPessimisticAutoTx TKey, TValue . For additional information about data access APIs, see the IBM.WebSphere.Caching.Map namespace documentation.
IBM.WebSphere.Caching.Map	The IBM.WebSphere.Caching.Map namespace includes the data access application programming interfaces. See the IBM.WebSphere.Caching namespace documentation for a description on how to access a map. The eXtreme Scale client supports transactional data access to individual maps using automatic and manual transactions. The following maps are available: • The IGridMapPessimisticAutoTx TKey, TValue interface provides automatic transactions. • The IGridMapPessimisticTx TKey, TValue interface provides manual transaction demarcation. See each respective interface for programming examples.
IBM.WebSphere.Caching.Security	The IBM.WebSphere.Caching.Security namespace includes the application programming interfaces specific to security. The eXtreme Scale client supports transport security and authenticiation and authorization using the ICredentialGenerator interface. Security is configured using a client properties file.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IBM.WebSphere.Caching namespace is the parent namespace for the the WebSphere eXtreme Scale C# client data access application programming interfaces.

The primary entry point for interacting with the data grid is the GridManagerFactory, which provides access to the IGridManager and IGrid interfaces.

The IGridManager interface provides methods to connect and disconnect from a catalog service domain. It also provides access to IGrid instances from the catalog service domain connections.

The IGrid interface allows clients to interact with a named data grid using various map interfaces, such as the IGridMapPessimisticTx TKey, TValue and IGridMapPessimisticAutoTx TKey, TValue .

For additional information about data access APIs, see the IBM.WebSphere.Caching.Map namespace documentation.

Classes

Class	Description
AvailabilityException	An AvailabilityException exception occurs when a target is not in the correct state to handle a request that it receives.
ClientServerTransactionCallbackException	A ClientServerTransactionCallbackException exception occurs when a method call to the client/server TransactionCallback encounters a remote request problem.
GridConfigurationException	An GridConfigurationException exception occurs when a configuration problem is found. This exception might occur when the configuration specified in the deployment policy, ObjectGrid descriptor, or security descriptor is not correct.
GridException	An GridException exception is the base exception class for all checked exceptions that occur in the product.
GridManagerFactory	The GridManagerFactory is a factory for IGridManager instances, and is the entrypoint for all interactions with the data grid.
GridServerRuntimeException	A GridServerRuntimeException exception is a generic wrapper for exceptions that occur in the server runtime.
LifecycleFailedException	A LifecycleFailedException exception occurs on unexpected lifecycle states.
MixedTransportException	A MixedTransportException is thrown when server and client have mismatched transport. For example, a server is using ORB, but the client is eXtremeIO
NoActiveTransactionException	A NoActiveTransactionException exception indicates that no active transactions exist.
NotReentrantException	A NotReentrantException occurs when a thread tries to run a map operation, such as calling a method on ObjectMap interface, when another thread is already running a map operation for the Session. A Session object can be used by a single thread only to perform concurrent map operations.
OrderedDictionary TKey, TValue	A generic version of the non-generic OrderedDictionary class.
OrderedDictionary TKey,	

TValue_Enumerator	The enumerator for interating through the OrderedDictionary TKey, TValue .
ReplicationVotedToRollbackTransactionException	A ReplicationVotedToRollbackTransactionException exception occurs when a transaction was rolled back because some or all of the synchronous replicas did not apply the transaction.
TransactionAlreadyActiveException	A TransactionAlreadyActiveException exception occurs to indicate that a transaction is already active for the current Session. This exception does not cause the current active transaction to be rolled back, so the isTransactionActive method returns true.
TransactionCallbackException	A TransactionCallbackException exception occurs when a TransactionCallback method call fails.
TransactionException	A TransactionException exception is a general locking exception that indicates something went wrong with a transaction. Use the isTransactionActive() and wasTransactionRolledBack() methods to determine whether transaction is still active or was rolled back as a result of this exception.
TransactionTimeoutException	A TransactionTimeoutException exception occurs when a transaction exceeds the transaction timeout value that was specified on the ObjectGrid or Session.

Interfaces

Interface	Description
ICatalogDomainInfo	Identifies a catalog service domain to be used for connecting to an eXtreme Scale data grid.
ICatalogDomainManager	The ICatalogDomainManager is a factory for ICatalogDomainInfo objects used to connect to a catalog service domain. Use the CatalogDomainManager to retrieve an ICatalogDomainMananager instance.
IClientConnectionContext	The handle to a connection to a catalog service domain. An IClientConnectionContext is returned from the Connect(ICatalogDomainInfo) method when connecting to a data grid. When finished with the data grid, use the Disconnect(IClientConnectionContext) method to disconnect from the data grid.
IGrid	An IGrid is the client's access to a named data grid and cooresponds to a configured ObjectGrid in the catalog service domain. An IGrid is thread safe and should be cached and reused by the application and is valid until the connection is severed. Use an IGrid to get map instances. Maps are defined on the data grid and have unique names within each configured ObjectGrid.
IGridManager	Provides methods for connecting to data grids. Use the GridManagerFactory to obtain an instance to an IGridManager instance.
IGridTransaction	Defines the interface for a transaction.
IOrderedDictionary TKey, TValue	Specifies a generic version of the non-generic IOrderedDictionary interface.
ITransactionable	Implementors of ITransactionable provide transaction demarcation semantics.

Enumerations

Enumer

ation	Description
AvailabilityState	Each shard in a distributed data has an associated availability state. This state determines if the shard can process incoming requests.
TxnIsolationLevel	Specifies an enumeration that defines the valid transaction isolation level values.

Remarks

Note: The client API in this namespace and child namespaces will evolve in future releases. Therefore, to avoid problems with incompatible bindings in the future, interfaces should not be directly implemented nor extended unless explicitly noted in the interface documentation.
Examples

The following example illustrates how to connect to a data grid, retrieve an entry from a map, and disconnect from the grid:
Copy to ClipboardPrint

```
// Retrieve the IGridManager instance. IGridManager gm = GridManagerFactory.GetGridManager(); // Id
```


Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An AvailabilityException exception occurs when a target is not in the correct state to handle a request that it receives.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching AvailabilityException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

AvailabilityException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AvailabilityException type exposes the following members.

Constructors

Name	Description
AvailabilityException	Initializes a new instance of the AvailabilityException class.
AvailabilityException(Exception)	Initializes a new instance of the AvailabilityException class with the specified exception cause.
AvailabilityException(String)	Initializes a new instance of the AvailabilityException class with the specified error message.
AvailabilityException(String, Exception)	Initializes a new instance of the AvailabilityException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	NewAvailabilityState	Gets or sets the availability state. This state controls if the shard can process incoming requests.
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

AvailabilityException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
AvailabilityException	Initializes a new instance of the AvailabilityException class.
AvailabilityException(Exception)	Initializes a new instance of the AvailabilityException class with the specified exception cause.
AvailabilityException(String)	Initializes a new instance of the AvailabilityException class with the specified error message.
AvailabilityException(String, Exception)	Initializes a new instance of the AvailabilityException class with the specified error message and exception cause.

Back to Top

See Also

AvailabilityException Class
AvailabilityException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that caused of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that caused of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AvailabilityException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

AvailabilityException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AvailabilityException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
NewAvailabilityState	Gets or sets the availability state. This state controls if the shard can process incoming requests.
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

AvailabilityException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the availability state. This state controls if the shard can process incoming requests.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

AvailabilityException Class
AvailabilityException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Each shard in a distributed data has an associated availability state. This state determines if the shard can process incoming requests.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Members

Member name	Value	Description
OFFLINE	0	An AvailabilityState.OFFLINE state indicates that a shard is offline and cannot process requests.
PRELOAD	1	An AvailabilityState.PRELOAD state indicates that a shard is in the preload state. When a shard is in the preload state, the shard allows requests only from a client that is preloading data into the data grid. All other requests are rejected.
ONLINE	2	An AvailabilityState.ONLINE state indicates that a shard is online. A shard that is online is available for processing requests.
QUIESCE	3	An AvailabilityState.QUIESCE state indicates that a shard is in quiesce. Quiesce is a transitional state. Shards that are in the quiesce state are being taken offline.
UNKNOWN	4	An AvailabilityState.UNKNOWN state indicates that the availability state of the shard could not be determined.

See Also

IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ClientServerTransactionCallbackException exception occurs when a method call to the client/server TransactionCallback encounters a remote request problem.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionCallbackException
        IBM.WebSphere.Caching ClientServerTransactionCallbackException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ClientServerTransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerTransactionCallbackException type exposes the following members.

Constructors

Name	Description
ClientServerTransactionCallbackException	Initializes a new instance of the ClientServerTransactionCallbackException class.
ClientServerTransactionCallbackException(Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified exception cause.
ClientServerTransactionCallbackException(String)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message.
ClientServerTransactionCallbackException(String, Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ClientServerTransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ClientServerTransactionCallbackException	Initializes a new instance of the ClientServerTransactionCallbackException class.
ClientServerTransactionCallbackException(Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified exception cause.
ClientServerTransactionCallbackException(String)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message.
ClientServerTransactionCallbackException(String, Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message and exception cause.

Back to Top

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerTransactionCallbackException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerTransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerTransactionCallbackException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerTransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An GridConfigurationException exception occurs when a configuration problem is found. This exception might occur when the configuration specified in the deployment policy, ObjectGrid descriptor, or security descriptor is not correct.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching GridConfigurationException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridConfigurationException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridConfigurationException type exposes the following members.

Constructors

Name	Description
GridConfigurationException	Initializes a new instance of the GridConfigurationException class.
GridConfigurationException(Exception)	Initializes a new instance of the GridConfigurationException class with a specified exception cause.
GridConfigurationException(String)	Initializes a new instance of the GridConfigurationException class with the specified error message.
GridConfigurationException(String, Exception)	Initializes a new instance of the GridConfigurationException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)

	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridConfigurationException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridConfigurationException	Initializes a new instance of the GridConfigurationException class.
GridConfigurationException(Exception)	Initializes a new instance of the GridConfigurationException class with a specified exception cause.
GridConfigurationException(String)	Initializes a new instance of the GridConfigurationException class with the specified error message.
GridConfigurationException(String, Exception)	Initializes a new instance of the GridConfigurationException class with the specified error message and exception cause.

Back to Top

See Also

GridConfigurationException Class
GridConfigurationException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

[Send Feedback]

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

[Send Feedback]

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridConfigurationException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

GridConfigurationException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridConfigurationException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridConfigurationException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

An GridException exception is the base exception class for all checked exceptions that occur in the product.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching AvailabilityException
      IBM.WebSphere.Caching GridConfigurationException
      IBM.WebSphere.Caching LifecycleFailedException
      IBM.WebSphere.Caching.Map CacheKeyNotFoundException
      IBM.WebSphere.Caching.Map DuplicateKeyException
      IBM.WebSphere.Caching.Map LoaderException
      IBM.WebSphere.Caching.Map LockException
      IBM.WebSphere.Caching.Map LockStrategyNotSupportedException
      IBM.WebSphere.Caching.Map MultiplePartitionWriteException
      IBM.WebSphere.Caching.Map OptimisticCollisionException
      IBM.WebSphere.Caching.Map ReadOnlyException
      IBM.WebSphere.Caching.Map TargetNotAvailableException
      IBM.WebSphere.Caching.Map UndefinedMapException
      IBM.WebSphere.Caching MixedTransportException
      IBM.WebSphere.Caching NotReentrantException
      IBM.WebSphere.Caching TransactionCallbackException
      IBM.WebSphere.Caching TransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridException Members
IBM.WebSphere.Caching Namespace

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridException type exposes the following members.

Constructors

Name	Description
GridException	Initializes a new instance of the GridException class.
GridException(Exception)	Initializes a new instance of the GridException class with a specified exception cause.
GridException(String)	Initializes a new instance of the GridException class with the specified error message.
GridException(String, Exception)	Initializes a new instance of the GridException class with the specified error message and exception cause.

Back to Top

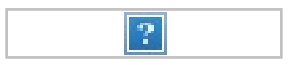
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

GridException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridException	Initializes a new instance of the GridException class.
GridException(Exception)	Initializes a new instance of the GridException class with a specified exception cause.
GridException(String)	Initializes a new instance of the GridException class with the specified error message.
GridException(String, Exception)	Initializes a new instance of the GridException class with the specified error message and exception cause.

Back to Top

See Also

GridException Class
GridException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridException type exposes the following members.
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top
See Also

GridException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridException type exposes the following members.
Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

GridException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridManagerFactory is a factory for IGridManager instances, and is the entrypoint for all interactions with the data grid.

Inheritance Hierarchy

System Object
  IBM.WebSphere.Caching GridManagerFactory

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridManagerFactory Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridManagerFactory type exposes the following members.

Methods

	Name	Description
	GetGridManager	Retrieves the IGridManager singleton instance that will be used by the client process to connect with the ObjectGrid.

Back to Top

See Also

GridManagerFactory Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridManagerFactory type exposes the following members.

Methods

	Name	Description
	GetGridManager	Retrieves the IGridManager singleton instance that will be used by the client process to connect with the ObjectGrid.

Back to Top

See Also

GridManagerFactory Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the IGridManager singleton instance that will be used by the client process to connect with the ObjectGrid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Return Value

The IGridManager singleton

See Also

GridManagerFactory Class
GridManagerFactory Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A GridServerRuntimeException exception is a generic wrapper for exceptions that occur in the server runtime.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridServerRuntimeException
      IBM.WebSphere.Caching.Security AccessControlException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridServerRuntimeException type exposes the following members.

Constructors

Name	Description
GridServerRuntimeException(String)	Initializes a new instance of the GridServerRuntimeException class.
GridServerRuntimeException(String, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message and the Java exception class name.
GridServerRuntimeException(String, Exception, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message, a reference to the inner exception that is the cause of this exception, and the Java exception class name.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Provides a user-readable representation of this GridServerRuntimeException exception. (Overrides Exception ToString .)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	JavaException ClassName	Identifies the Java Exception class name that this exception identifies.
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

GridServerRuntimeException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridServerRuntimeException(String)	Initializes a new instance of the GridServerRuntimeException class.
GridServerRuntimeException(String, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message and the Java exception class name.
GridServerRuntimeException(String, Exception, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message, a reference to the inner exception that is the cause of this exception, and the Java exception class name.

Back to Top

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridServerRuntimeException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

javaExceptionClassName
Type: System String
Specifies the Java exception class name.

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
GridServerRuntimeException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridServerRuntimeException class with the specified error message and the Java exception class name.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

javaExceptionClassName
Type: System String
Specifies the Java exception class name.

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
GridServerRuntimeException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridServerRuntimeException class with the specified error message, a reference to the inner exception that is the cause of this exception, and the Java exception class name.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

innerException
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

javaExceptionClassName
Type: System String
Specifies the Java exception class name.

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
GridServerRuntimeException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridServerRuntimeException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Provides a user-readable representation of this GridServerRuntimeException exception. (Overrides Exception ToString .)

Back to Top

See Also

GridServerRuntimeException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Provides a user-readable representation of this GridServerRuntimeException exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

The user-readable representation of this exception.

Implements

_Exception ToString

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

The GridServerRuntimeException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
JavaExceptionClassName	Identifies the Java Exception class name that this exception identifies.
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridServerRuntimeException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Identifies the Java Exception class name that this exception identifies.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Identifies a catalog service domain to be used for connecting to an eXtreme Scale data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ICatalogDomainInfo Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainInfo type exposes the following members.

Properties

Name	Description
CatalogServerAddresses	Retrieves the configured catalog service addresses. See the CreateCatalogDomainInfo(String) method for more details of the format of the catalog service address string.

Back to Top

See Also

ICatalogDomainInfo Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainInfo type exposes the following members.

Properties

Name	Description
CatalogServerAddresses	Retrieves the configured catalog service addresses. See the CreateCatalogDomainInfo(String) method for more details of the format of the catalog service address string.

Back to Top

See Also

ICatalogDomainInfo Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the configured catalog service addresses. See the CreateCatalogDomainInfo(String) method for more details of the format of the catalog service address string.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICatalogDomainInfo Interface
ICatalogDomainInfo Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainManager is a factory for ICatalogDomainInfo objects used to connect to a catalog service domain. Use the CatalogDomainManager to retrieve an ICatalogDomainMananager instance.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICatalogDomainManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainManager type exposes the following members.

Methods

Name	Description
CreateCatalogDomainInfo	Creates an instance of the ICatalogDomainInfo object for the specified catalog service addresses. The catalog service addresses are expressed in a string of the form: catalogServerAddresses ::= <catalogServiceEndpoint> [, <catalogServiceEndpoint>] catalogServiceEndpoint ::= <hostName> : <listenerPort> hostName ::= The IP address or host name of a catalog service. listenerPort ::= The listener port that the catalog service is configured to use. For example: com.acme.server1:2809,com.acme.server2:2809

Back to Top

See Also

ICatalogDomainManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainManager type exposes the following members.

Methods

Name	Description
CreateCatalogDomainInfo	Creates an instance of the ICatalogDomainInfo object for the specified catalog service addresses. The catalog service addresses are expressed in a string of the form: catalogServerAddresses ::= <catalogServiceEndpoint> [, <catalogServiceEndpoint>] catalogServiceEndpoint ::= <hostName> : <listenerPort> hostName ::= The IP address or host name of a catalog service. listenerPort ::= The listener port that the catalog service is configured to use. For example: com.acme.server1:2809,com.acme.server2:2809

Back to Top

See Also

ICatalogDomainManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Creates an instance of the ICatalogDomainInfo object for the specified catalog service addresses.

The catalog service addresses are expressed in a string of the form:

catalogServerAddresses ::= <catalogServiceEndpoint> [,<catalogServiceEndpoint>]

catalogServiceEndpoint ::= <hostName> : <listenerPort>

hostName ::= The IP address or host name of a catalog service.

listenerPort ::= The listener port that the catalog service is configured to use.

For example:

com.acme.server1:2809,com.acme.server2:2809

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

catalogServerAddresses
Type: System String
The catalog service addresses.

Return Value

The ICatalogDomainInfo representing the catalog service domain addresses.
See Also

ICatalogDomainManager Interface
ICatalogDomainManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The handle to a connection to a catalog service domain. An IClientConnectionContext is returned from the Connect(ICatalogDomainInfo) method when connecting to a data grid.

When finished with the data grid, use the Disconnect(IClientConnectionContext) method to disconnect from the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IClientConnectionContext Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IClientConnectionContext type exposes the following members.

Properties

Name	Description
DomainName	Retrieves the name of the catalog service domain.

Back to Top

See Also

IClientConnectionContext Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IClientConnectionContext type exposes the following members.

Properties

Name	Description
DomainName	Retrieves the name of the catalog service domain.

Back to Top

See Also

IClientConnectionContext Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the name of the catalog service domain.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

IClientConnectionContext Interface
IClientConnectionContext Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IGrid Interface IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An IGrid is the client's access to a named data grid and cooresponds to a configured ObjectGrid in the catalog service domain.

An IGrid is thread safe and should be cached and reused by the application and is valid until the connection is severed.

Use an IGrid to get map instances. Maps are defined on the data grid and have unique names within each configured ObjectGrid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

The IGrid type exposes the following members.

Methods

Name	Description
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
GetGridMapNames	Returns a list of map names that are associated with this data grid.
GetGridMapPessimisticAutoTx TKey, TValue (String)	Returns a map instance for the specified map.
GetGridMapPessimisticAutoTx TKey, TValue (String, ICredentialGenerator)	Returns a map instance for the specified map using specified credential object (userid password).
GetGridMapPessimisticTx TKey, TValue (String)	Returns a grid map instance for the specified map.
GetGridMapPessimisticTx TKey, TValue (String, ICredentialGenerator)	Returns a grid map instance for the specified map using specified credential object (userid password)

Back to Top

Properties

Name	Description
IsSecurityEnabled	Retreives a value that indicates whether data grid security is enabled.
Name	Retrieves the data grid name.
TransactionTimeout	Gets or sets the default transaction timeout for the data grid.

Back to Top

See Also

IGrid Interface
IBM.WebSphere.Caching Namespace

The IGrid type exposes the following members.

Methods

Name	Description
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
GetGridMapNames	Returns a list of map names that are associated with this data grid.
GetGridMapPessimisticAutoTx TKey, TValue (String)	Returns a map instance for the specified map.
GetGridMapPessimisticAutoTx TKey, TValue (String, ICredentialGenerator)	Returns a map instance for the specified map using specified credential object (userid password).
GetGridMapPessimisticTx TKey, TValue (String)	Returns a grid map instance for the specified map.
GetGridMapPessimisticTx TKey, TValue (String, ICredentialGenerator)	Returns a grid map instance for the specified map using specified credential object (userid password)

Back to Top

See Also

IGrid Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a list of map names that are associated with this data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Returns a list of map names.

See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

	Name	Description
	GetGridMapPessimisticAutoTx TKey, TValue (String)	Returns a map instance for the specified map.
	GetGridMapPessimisticAutoTx TKey, TValue (String, ICredentialGenerator)	Returns a map instance for the specified map using specified credential object (userid password).

Back to Top

See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map instance for the specified map.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

mapName
: Type: System String
Specifies the map name.

Type Parameters

TKey
: Specifies a generic type key.

TValue
: Specifies a generic type Value.

Return Value

An IGridMapPessimisticAutoTx<TKey,TValue> instance

Exceptions

Exception	Condition
IBM.WebSphere.Caching.MapLockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not supported by this map interface.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticAutoTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map instance for the specified map using specified credential object (userid password).

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

mapName
Type: System String
Specifies the map name.

credentialGenerator
Type: IBM.WebSphere.Caching.Security ICredentialGenerator
Specifies the Credential Generator object.

Type Parameters

TKey
Specifies a generic type key.
TValue
Specifies a generic type Value.

Return Value

An IGridMapPessimisticAutoTx<TKey,TValue> instance
Exceptions

Exception	Condition
IBM.WebSphere.Caching.Map LockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not supported by this map interface.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticAutoTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GetGridMapPessimisticTx TKey, TValue (String)	Returns a grid map instance for the specified map.
GetGridMapPessimisticTx TKey, TValue (String, ICredentialGenerator)	Returns a grid map instance for the specified map using specified credential object (userid password)

Back to Top

See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a grid map instance for the specified map.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

mapName
: Type: System String
: mapName is the name of the specified map

Type Parameters

TKey
: Generic type key.

TValue
: Generic Type Value.

Return Value

An IGridMapPessimisticTx<TKey,TValue> instance

Exceptions

Exception	Condition
IBM.WebSphere.Caching.MapLockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not configured for pessimistic locking.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a grid map instance for the specified map using specified credential object (userid password)

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

mapName
Type: System String
mapName is the name of the specified map

credentialGenerator
Type: IBM.WebSphere.Caching.Security ICredentialGenerator
Specifies the Credential Generator object.

Type Parameters

TKey
Generic type key.
TValue
Generic Type Value.

Return Value

An IGridMapPessimisticTx<TKey,TValue> instance
Exceptions

Exception	Condition
IBM.WebSphere.Caching.Map LockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not configured for pessimistic locking.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGrid type exposes the following members.
Properties

	Name	Description
	IsSecurityEnabled	Retreives a value that indicates whether data grid security is enabled.
	Name	Retrieves the data grid name.
	TransactionTimeout	Gets or sets the default transaction timeout for the data grid.

Back to Top
See Also

IGrid Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retreives a value that indicates whether data grid security is enabled.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if data grid security is enabled; otherwise, false.
See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the data grid name.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The data grid name.
See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the default transaction timeout for the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The default transaction timeout value for the data grid.
See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IGridManager Interface IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Provides methods for connecting to data grids. Use the GridManagerFactory to obtain an instance to an IGridManager instance.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Examples

The following example illustrates how to retrieve an IGridManager and disconnect:
Copy to ClipboardPrint

```
// Retrieve the singleton IGridManager instance.
IGridManager gm = GridManagerFactory.GetGridManager();

// Retreive and cache the grid, and retreive and update data....
...

// Disconnect from the data grid when all done and
// null out any references.
gm.Disconnect(ccc);
grid = null;
ccc = null;
```

See Also

IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IGridManager Members IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridManager type exposes the following members.

Methods

Name	Description
Connect(ICatalogDomainInfo)	Connects a client process to the data grid. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Connect(ICatalogDomainInfo, String)	Connects a client process to the data grid using connection properties that are specified in the the configFilename. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Disconnect	Disconnects a client process from a catalog server. The IClientConnectionContext and associated IGrid instances are invalid once disconnected.
GetGrid	Returns a client data grid instance that corresponds to the data grid that is identified by the specified name. The IGrid is thread safe and should be cached for the life of the application. A grid name cooresponds to an ObjectGrid name in the data grid configuration.

Back to Top

Properties

Name	Description
CatalogDomainManager	Retrieves the CatalogDomainManager, which is used for creating ICatalogDomainInfo objects.

Back to Top

See Also

IGridManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridManager type exposes the following members.
Methods

Name	Description
Connect(ICatalogDomainInfo)	Connects a client process to the data grid. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Connect(ICatalogDomainInfo, String)	Connects a client process to the data grid using connection properties that are specified in the the configFilename. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Disconnect	Disconnects a client process from a catalog server. The IClientConnectionContext and associated IGrid instances are invalid once disconnected.
GetGrid	Returns a client data grid instance that corresponds to the data grid that is identified by the specified name. The IGrid is thread safe and should be cached for the life of the application. A grid name cooresponds to an ObjectGrid name in the data grid configuration.

Back to Top
See Also

IGridManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Connect(ICatalogDomainInfo)	Connects a client process to the data grid. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Connect(ICatalogDomainInfo, String)	Connects a client process to the data grid using connection properties that are specified in the the configFilename. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.

Back to Top

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Connects a client process to the data grid.

Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

catalogDomainInfo
Type: IBM.WebSphere.Caching ICatalogDomainInfo
Specifies the CatalogDomainInfo object that contains target catalog service domain information.

Return Value

Returns an IClientConnectionContext object that represents a handle to the data grid to which the client is connected.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullcatalogDomainInfo is specified.
IBM.WebSphere.Caching GridException	Occurs if there is an error connecting to the specified catalog domain.

See Also

IGridManager Interface
IGridManager Members
Connect Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Connects a client process to the data grid using connection properties that are specified in the the configFilename.

Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

catalogDomainInfo
Type: IBM.WebSphere.Caching ICatalogDomainInfo
Specifies the CatalogDomainInfo object that contains target catalog service domain information.

configFilename
Type: System String
Specifies the full path and file name of the client configuration file or null to use the default configuration.

Return Value

Returns an IClientConnectionContext object that represents a handle to the data grid to which the client is connected.
Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullcatalogDomainInfo is specified.
IBM.WebSphere.Caching GridException	Occurs if there is an error connecting to the specified catalog domain.
System.IO FileNotFoundException	Occurs if configFilename does not exist.

See Also

IGridManager Interface
IGridManager Members
Connect Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Disconnects a client process from a catalog server. The IClientConnectionContext and associated IGrid instances are invalid once disconnected.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

context
: Type: IBM.WebSphere.Caching IClientConnectionContext
Specifies the IClientConnectionContext object that was returned from a previous Connect method call.

Return Value

Returns true if the disconnect was successful, or false if the supplied context was not connected.

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a client data grid instance that corresponds to the data grid that is identified by the specified name. The IGrid is thread safe and should be cached for the life of the application.

A grid name cooresponds to an ObjectGrid name in the data grid configuration.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

context
: Type: IBM.WebSphere.Caching IClientConnectionContext
: Specifies the IClientConnectionContext object returned from a Connect method call.

gridName
: Type: System String
: Specifies the name of the requested ObjectGrid.

Return Value

Returns client data grid instance that corresponds to the specified remote data grid.
Exceptions

Exception	Condition
IBM.WebSphere.Caching GridException	Occurs if the specified grid is not found or incompatible with the current configuration.

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridManager type exposes the following members.
Properties

	Name	Description
	CatalogDomainManager	Retrieves the CatalogDomainManager, which is used for creating ICatalogDomainInfo objects.

Back to Top
See Also

IGridManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the CatalogDomainManager, which is used for creating ICatalogDomainInfo objects.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Defines the interface for a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridTransaction type exposes the following members.

Methods

Name	Description
Begin	Begins a new transaction.
Commit	Commits a transaction.
Flush	Apply the current changes to the server without committing the transaction.
MarkRollbackOnly	Marks the current transaction as being rollback only.
Rollback	Rolls back a transaction.

Back to Top

Properties

Name	Description
Active	Determines if a transaction is currently active.
MarkedRollbackOnly	Determines if a transaction is roll back only.
TransactionIsolationLevel	Gets or sets the transaction isolation level.
TransactionTimeout	Gets or sets the amount of time in which a transaction must complete. To use an unlimited transaction timeout value, set the value to TimeSpan.Zero

Back to Top

See Also

IGridTransaction Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridTransaction type exposes the following members.

Methods

Name	Description
Begin	Begins a new transaction.
Commit	Commits a transaction.
Flush	Apply the current changes to the server without committing the transaction.
MarkRollbackOnly	Marks the current transaction as being rollback only.
Rollback	Rolls back a transaction.

Back to Top

See Also

IGridTransaction Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Begins a new transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Commits a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Apply the current changes to the server without committing the transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Marks the current transaction as being rollback only.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

error
Type: System Exception
Specifies the cause of the exception.

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Rolls back a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridTransaction type exposes the following members.

Properties

Name	Description
Active	Determines if a transaction is currently active.
MarkedRollbackOnly	Determines if a transaction is roll back only.
TransactionIsolationLevel	Gets or sets the transaction isolation level.
TransactionTimeout	Gets or sets the amount of time in which a transaction must complete. To use an unlimited transaction timeout value, set the value to TimeSpan.Zero

Back to Top

See Also

IGridTransaction Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines if a transaction is currently active.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines if a transaction is roll back only.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the transaction isolation level.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The transaction isolation level.

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the amount of time in which a transaction must complete. To use an unlimited transaction timeout value, set the value to TimeSpan.Zero

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The transaction timeout value.
See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies a generic version of the non-generic IOrderedDictionary interface.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Type Parameters

TKey
: Specifies the key type.

TValue
: Specifies the value type.

See Also

IOrderedDictionary TKey, TValue _Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IOrderedDictionary TKey, TValue type exposes the following members.

Methods

Name	Description
Add(T)	Adds an item to the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Add(TKey, TValue)	Adds an element with the provided key and value to the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Clear	Removes all items from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Contains	Determines whether the ICollection T contains a specific value. (Inherited from ICollection KeyValuePair TKey, TValue .)
ContainsKey	Determines whether the IDictionary TKey, TValue contains an element with the specified key. (Inherited from IDictionary TKey, TValue .)
CopyTo	Copies the elements of the ICollection T to an Array, starting at a particular Array index. (Inherited from ICollection KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through the collection. (Inherited from IEnumerable KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through a collection. (Inherited from IEnumerable.)
Insert	Inserts an item with the specified key and value into the map at the specified index.
Remove(TKey)	Removes the element with the specified key from the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Remove(T)	Removes the first occurrence of a specific object from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
RemoveAt	Removes the item at the specified index from the IOrderedDictionary object.
TryGetValue	Gets the value associated with the specified key. (Inherited from IDictionary TKey, TValue .)

Back to Top

Properties

Name	Description
Count	Gets the number of elements contained in the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
IsReadOnly	Gets a value indicating whether the ICollection T is read-only. (Inherited from ICollection KeyValuePair TKey, TValue .)
Item TKey	Gets or sets the element with the specified key. (Inherited from IDictionary TKey, TValue .)
Item Int32	Gets or sets the value with the specified index.
Keys	Gets an ICollection T containing the keys of the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)

Values	Gets an ICollection T containing the values in the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)

Back to Top
See Also

IOrderedDictionary TKey, TValue Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IOrderedDictionary TKey, TValue type exposes the following members.

Methods

Name	Description
Add(T)	Adds an item to the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Add(TKey, TValue)	Adds an element with the provided key and value to the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Clear	Removes all items from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Contains	Determines whether the ICollection T contains a specific value. (Inherited from ICollection KeyValuePair TKey, TValue .)
ContainsKey	Determines whether the IDictionary TKey, TValue contains an element with the specified key. (Inherited from IDictionary TKey, TValue .)
CopyTo	Copies the elements of the ICollection T to an Array, starting at a particular Array index. (Inherited from ICollection KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through the collection. (Inherited from IEnumerable KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through a collection. (Inherited from IEnumerable.)
Insert	Inserts an item with the specified key and value into the map at the specified index.
Remove(TKey)	Removes the element with the specified key from the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Remove(T)	Removes the first occurrence of a specific object from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
RemoveAt	Removes the item at the specified index from the IOrderedDictionary object.
TryGetValue	Gets the value associated with the specified key. (Inherited from IDictionary TKey, TValue .)

Back to Top

See Also

IOrderedDictionary TKey, TValue Interface
IBM.WebSphere.Caching Namespace

Inserts an item with the specified key and value into the map at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
Specifies the zero-based index where the item should be inserted

key
Type: TKey
Specifies the key of the item to insert.

value
Type: TValue
Specifies the value of the item to insert.

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Occurs when index is out of range.
System ArgumentNullException	Occurs when a null key is specified.
System ArgumentException	Occurs when key already exists in the IOrderedDictionary object.

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the item at the specified index from the IOrderedDictionary object.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index of the item to remove.

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Occurs when index is less than 0 or greater than Count value.

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IOrderedDictionary TKey, TValue type exposes the following members.

Properties

	Name	Description
	Count	Gets the number of elements contained in the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
	IsReadOnly	Gets a value indicating whether the ICollection T is read-only. (Inherited from ICollection KeyValuePair TKey, TValue .)
	Item TKey	Gets or sets the element with the specified key. (Inherited from IDictionary TKey, TValue .)
	Item Int32	Gets or sets the value with the specified index.
	Keys	Gets an ICollection T containing the keys of the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
	Values	Gets an ICollection T containing the values in the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)

Back to Top

See Also

IOrderedDictionary TKey, TValue Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

	Name	Description
	Item TKey	Gets or sets the element with the specified key. (Inherited from IDictionary TKey, TValue .)
	Item Int32	Gets or sets the value with the specified index.

Back to Top

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Gets or sets the value with the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

index
Type: System Int32
Specifies the zero-based index of the value to get or set.

Return Value

The value of the item at the specified index.

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Occurs when index is less than 0 or greater than the Count value.

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
Item Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Implementors of ITransactionable provide transaction demarcation semantics.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ITransactionable Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ITransactionable type exposes the following members.

Properties

Name	Description
Transaction	The Transaction instance used to configure and demarcate a transaction.

Back to Top

See Also

ITransactionable Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ITransactionable type exposes the following members.

Properties

Name	Description
Transaction	The Transaction instance used to configure and demarcate a transaction.

Back to Top

See Also

ITransactionable Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The Transaction instance used to configure and demarcate a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ITransactionable Interface
ITransactionable Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LifecycleFailedException exception occurs on unexpected lifecycle states.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching LifecycleFailedException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LifecycleFailedException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LifecycleFailedException type exposes the following members.

Constructors

Name	Description
LifecycleFailedException(String)	Initializes a new instance of the LifecycleFailedException class with the specified error message.
LifecycleFailedException(String, Exception)	Initializes a new instance of the LifecycleFailedException class with the specified error message and exception cause.

Back to Top

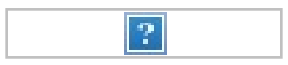
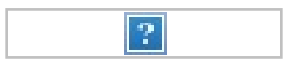
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)

	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LifecycleFailedException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LifecycleFailedException(String)	Initializes a new instance of the LifecycleFailedException class with the specified error message.
LifecycleFailedException(String, Exception)	Initializes a new instance of the LifecycleFailedException class with the specified error message and exception cause.

Back to Top

See Also

LifecycleFailedException Class
LifecycleFailedException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LifecycleFailedException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LifecycleFailedException Class
LifecycleFailedException Members
LifecycleFailedException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LifecycleFailedException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the exception message text.

cause
Type: System Exception
Specifies the initial cause of the exception that caused this exception to occur.

See Also

LifecycleFailedException Class
LifecycleFailedException Members
LifecycleFailedException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LifecycleFailedException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LifecycleFailedException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LifecycleFailedException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LifecycleFailedException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A MixedTransportException is thrown when server and client have mismatched transport. For example, a server is using ORB, but the client is eXtremeIO

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching MixedTransportException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

MixedTransportException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MixedTransportException type exposes the following members.

Constructors

Name	Description
MixedTransportException	Initializes a new instance of the MixedTransportException class.
MixedTransportException(String)	Initializes a new instance of the MixedTransportException class with the specified error message.

Back to Top

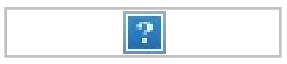
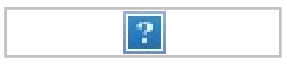
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)

	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

MixedTransportException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
MixedTransportException	Initializes a new instance of the MixedTransportException class.
MixedTransportException(String)	Initializes a new instance of the MixedTransportException class with the specified error message.

Back to Top

See Also

MixedTransportException Class
MixedTransportException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MixedTransportException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

MixedTransportException Class
MixedTransportException Members
MixedTransportException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MixedTransportException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

MixedTransportException Class
MixedTransportException Members
MixedTransportException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MixedTransportException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

MixedTransportException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MixedTransportException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

MixedTransportException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A NoActiveTransactionException exception indicates that no active transactions exist.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionException
        IBM.WebSphere.Caching NoActiveTransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

NoActiveTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Constructors

Name	Description
NoActiveTransactionException	Initializes a new instance of the NoActiveTransactionException class.
NoActiveTransactionException(Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified exception cause.
NoActiveTransactionException(String)	Initializes a new instance of the NoActiveTransactionException class with the specified error message.
NoActiveTransactionException(String, Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

Fields

Name	Description
ivTransactionRolledBa	Indicates whether the transaction was rolled back or

ck	not. (Inherited from TransactionException.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
NoActiveTransactionException	Initializes a new instance of the NoActiveTransactionException class.
NoActiveTransactionException(Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified exception cause.
NoActiveTransactionException(String)	Initializes a new instance of the NoActiveTransactionException class with the specified error message.
NoActiveTransactionException(String, Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified error message and exception cause.

Back to Top

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not. (Inherited from TransactionException.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A NotReentrantException occurs when a thread tries to run a map operation, such as calling a method on ObjectMap interface, when another thread is already running a map operation for the Session. A Session object can be used by a single thread only to perform concurrent map operations.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching NotReentrantException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

NotReentrantException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NotReentrantException type exposes the following members.

Constructors

Name	Description
NotReentrantException	Initializes a new instance of the NotReentrantException class.
NotReentrantException(Exception)	Initializes a new instance of the NotReentrantException class with a specified exception cause.
NotReentrantException(String)	Initializes a new instance of the NotReentrantException class with the specified error message.
NotReentrantException(String, Exception)	Initializes a new instance of the NotReentrantException class with the specified error message and exception cause.

Back to Top

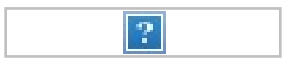
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception.

		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NotReentrantException Class

IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
NotReentrantException	Initializes a new instance of the NotReentrantException class.
NotReentrantException(Exception)	Initializes a new instance of the NotReentrantException class with a specified exception cause.
NotReentrantException(String)	Initializes a new instance of the NotReentrantException class with the specified error message.
NotReentrantException(String, Exception)	Initializes a new instance of the NotReentrantException class with the specified error message and exception cause.

Back to Top

See Also

NotReentrantException Class
NotReentrantException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NotReentrantException type exposes the following members.

Methods

	Name	Description
	Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
	Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
	GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
	GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

NotReentrantException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NotReentrantException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NotReentrantException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A generic version of the non-generic OrderedDictionary class.

Inheritance Hierarchy

System Object
  IBM.WebSphere.Caching OrderedDictionary TKey, TValue

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
    The key type
TValue
    The value type

See Also

OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue type exposes the following members.

Constructors

Name	Description
OrderedDictionary TKey, TValue	Initializes a new instance of the OrderedDictionary TKey, TValue class.
OrderedDictionary TKey, TValue (Int32)	Initializes a new instance of the OrderedDictionary TKey, TValue class with the specified capacity.
OrderedDictionary TKey, TValue (IOrderedDictionary TKey, TValue)	Initializes a new instance of the OrderedDictionary TKey, TValue class having its contents copied from the specified IOrderedDictionary TKey, TValue .

Back to Top

Methods

Name	Description
Add(KeyValuePair TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Add(TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Clear	Removes all items from the OrderedDictionary TKey, TValue .
Contains	Determines whether the OrderedDictionary TKey, TValue contains the specified item.
ContainsKey	Determines whether the OrderedDictionary TKey, TValue contains the specified key.
CopyTo(Array, Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.
CopyTo(KeyValue Pair TKey, TValue , Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetEnumerator	Returns a Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
Insert(Int32, KeyValuePair TKey , TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
Insert(Int32, TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)

Remove(KeyValuePair TKey, TValue)	Removes the specified item from the OrderedDictionary TKey, TValue .
Remove(TKey)	Removes the item with the specified key from the OrderedDictionary TKey, TValue .
RemoveAt	Removes the item at the specified index from the OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)
TryGetValue	Get the value associated with the specified key.

Back to Top

Properties

Name	Description
Count	Gets the number of items in the OrderedDictionary TKey, TValue .
IsFixedSize	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue has a fixed size.
IsReadOnly	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue is read only.
IsSynchronized	Gets a value indicating whether access to this Collection is synchronized (thread safe).
Item Int32	Gets or sets the value associated with the specified index.
Item TKey	Gets or sets the value associated with the specified key.
Keys	Gets a ICollection T object containing all the keys in the OrderedDictionary TKey, TValue .
SyncRoot	Gets an object that can be used to synchronize this Collection.
Values	Gets a ICollection T object containing all the values in the OrderedDictionary TKey, TValue .

Back to Top

See Also

OrderedDictionary TKey, TValue Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
OrderedDictionary TKey, TValue	Initializes a new instance of the OrderedDictionary TKey, TValue class.
OrderedDictionary TKey, TValue (Int32)	Initializes a new instance of the OrderedDictionary TKey, TValue class with the specified capacity.
OrderedDictionary TKey, TValue (IOrderedDictionary TKey, TValue)	Initializes a new instance of the OrderedDictionary TKey, TValue class having its contents copied from the specified IOrderedDictionary TKey, TValue.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OrderedDictionary TKey, TValue class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
OrderedDictionary TKey, TValue Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OrderedDictionary TKey, TValue class with the specified capacity.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

capacity
Type: System Int32
The initial number of elements that OrderedDictionary TKey, TValue can contain.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
OrderedDictionary TKey, TValue Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OrderedDictionary TKey, TValue class having its contents copied from the specified IOrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

orderedDictionary
Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
The IOrderedDictionary TKey, TValue whose contents are copied into the new OrderedDictionary TKey, TValue .

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
OrderedDictionary TKey, TValue Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue type exposes the following members.

Methods

Name	Description
Add(KeyValuePair TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Add(TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Clear	Removes all items from the OrderedDictionary TKey, TValue .
Contains	Determines whether the OrderedDictionary TKey, TValue contains the specified item.
ContainsKey	Determines whether the OrderedDictionary TKey, TValue contains the specified key.
CopyTo(Array, Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.
CopyTo(KeyValue Pair TKey, TValue , Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetEnumerator	Returns a Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
Insert(Int32, KeyValuePair TKey , TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
Insert(Int32, TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
Remove(KeyValue Pair TKey, TValue)	Removes the specified item from the OrderedDictionary TKey, TValue .
Remove(TKey)	Removes the item with the specified key from the OrderedDictionary TKey, TValue .
RemoveAt	Removes the item at the specified index from the OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)
TryGetValue	Get the value associated with the specified key.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Add(KeyValuePair TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Add(TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The key-value pair to add.

Implements

ICollection T Add(T)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.
System.Collections.Generic KeyNotFoundException	Thrown when the kvp key is not found.

Remarks

Use this property to add value in the collection. Values are appended to the end of the collection. A key cannot be null; however a value can be. To update an existing item, use either the Item Int32 or the Item TKey property.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Add Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the item to add.

value
Type: TValue
The value of the item to add.

Implements

IDictionary TKey, TValue Add(TKey, TValue)
Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

Remarks

Use this property to add a new item to the collection. Items are appended to the end of the collection. A key cannot be null; however a value can be. To update an existing item, use either the Item Int32 or the Item Int32 property.
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Add Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes all items from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Implements

ICollection T Clear
IDictionary Clear
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines whether the OrderedDictionary TKey, TValue contains the specified item.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The key-value pair to locate in the OrderedDictionary TKey, TValue .

Return Value

true if the OrderedDictionary TKey, TValue contains the item associated with the specified kvp; otherwise, false.

Implements

ICollection T Contains(T)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.

Remarks

The specified item's key must be contained in the OrderedDictionary TKey, TValue and it's associated value must be equal to the specified item's value for true to be returned.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines whether the OrderedDictionary TKey, TValue contains the specified key.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

key
Type: TKey
The key to locate in the OrderedDictionary TKey, TValue .

Return Value

true if the OrderedDictionary TKey, TValue contains the item associated with the specified key; otherwise, false.

Implements

IDictionary TKey, TValue ContainsKey(TKey)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
CopyTo(Array, Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.
CopyTo(KeyValuePair TKey, TValue , Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

array
Type: System Array
The allocated single-dimension array into which the items from OrderedDictionary TKey, TValue will be copied.

index
Type: System Int32
The index into array where copying will begin.

Implements

ICollection CopyTo(Array, Int32)

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is out of range.
System ArgumentException	Thrown when array does not have sufficient size to hold the incoming data.

Remarks

The array must have sufficient storage to contain all items from the OrderedDictionary TKey, TValue ; otherwise a ArgumentException will be thrown. index must be equal or greater than 0.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
CopyTo Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

array
Type: System.Collections.Generic KeyValuePair TKey, TValue
The allocated array into which the items from OrderedDictionary TKey, TValue will be copied.

index
Type: System Int32
The zero-based index into array where copying will begin.

Implements

ICollection T CopyTo(T , Int32)

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is out of range.
System ArgumentException	Thrown when array does not have sufficient size to hold the incoming data.

Remarks

The array must have sufficient storage to contain all items from the OrderedDictionary TKey, TValue ; othewise a ArgumentException will be thrown. index must be equal or greater than 0.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
CopyTo Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

A Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .

Implements

IEnumerable T GetEnumerator
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Insert(Int32, KeyValuePair TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
Insert(Int32, TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index where the item should be inserted.

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The item to insert.

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.
System ArgumentOutOfRangeException	Thrown when index is out of range.

Remarks

The index must be greater than or equal to 0 and the index must be less than the number of elements in the collection; otherwise a ArgumentOutOfRangeException is thrown.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Insert Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index where the item should be inserted.

key
Type: TKey
The key of the item to insert.

value
Type: TValue
The value of the item to insert.

Implements

IOrderedDictionary TKey, TValue Insert(Int32, TKey, TValue)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the key is null.
System ArgumentOutOfRangeException	Thrown when index is out of range.

Remarks

The index must be greater than or equal to 0 and the index must be less than the number of elements in the collection; otherwise a ArgumentOutOfRangeException is thrown.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Insert Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Remove(KeyValuePair TKey, TValue)	Removes the specified item from the OrderedDictionary TKey, TValue .
Remove(TKey)	Removes the item with the specified key from the OrderedDictionary TKey, TValue .

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the specified item from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The key-value pair to remove.

Return Value

true if the item is successfully removed; otherwise false is returned. false is also returned if the item is not found.

Implements

ICollection T Remove(T)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Remove Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the item with the specified key from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the item to remove.

Return Value

true if the item is successfully removed; otherwise false. false is also returned if the key is not found.

Implements

IDictionary TKey, TValue Remove(TKey)
Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Remove Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the item at the specified index from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index of the item to remove.

Implements

IOrderedDictionary TKey, TValue _RemoveAt(Int32)

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is out of range.

Remarks

The index must be greater than or equal to 0 and the index must be less than the number of elements in the collection; otherwise a ArgumentOutOfRangeException is thrown.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Get the value associated with the specified key.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

key
Type: TKey
The key of the value to get.

val
Type: TValue
The value associated with the specified key.

Return Value

true if the OrderedDictionary TKey, TValue contained the specified key; otherwise false.

Implements

IDictionary TKey, TValue TryGetValue(TKey, TValue)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

Remarks

If the key is not found, val is initialized to the default value associated with its data type.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue type exposes the following members.

Properties

	Name	Description
	Count	Gets the number of items in the OrderedDictionary TKey, TValue.
	IsFixedSize	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue has a fixed size.
	IsReadOnly	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue is read only.
	IsSynchronized	Gets a value indicating whether access to this Collection is synchronized (thread safe).
	Item Int32	Gets or sets the value associated with the specified index.
	Item TKey	Gets or sets the value associated with the specified key.
	Keys	Gets a ICollection T object containing all the keys in the OrderedDictionary TKey, TValue.
	SyncRoot	Gets an object that can be used to synchronize this Collection.
	Values	Gets a ICollection T object containing all the values in the OrderedDictionary TKey, TValue.

Back to Top

See Also

OrderedDictionary TKey, TValue Class

IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the number of items in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The number of items in the OrderedDictionary TKey, TValue .

Implements

ICollection T Count
ICollection Count
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Gets a value that indicates whether or not the OrderedDictionary TKey, TValue has a fixed size.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if the OrderedDictionary TKey, TValue has a fixed size; otherwise false.

Implements

IDictionary IsFixedSize
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets a value that indicates whether or not the OrderedDictionary TKey, TValue is read only.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if the OrderedDictionary TKey, TValue is read only; otherwise false.

Implements

ICollection T IsReadOnly
IDictionary IsReadOnly
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets a value indicating whether access to this Collection is synchronized (thread safe).

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if this Collection is synchronized.

Implements

ICollection IsSynchronized

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Item Int32	Gets or sets the value associated with the specified index.
Item TKey	Gets or sets the value associated with the specified key.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Gets or sets the value associated with the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index of the value to get or set.

Field Value

The value of the item at the specified index.

Implements

IOrderedDictionary TKey, TValue Item Int32
Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is less than 0 or greater than the number of elements in the collection.

Remarks

A set operation will add or replace a value in the collection, with new values being appended to the end of the collection. A key must be greater than or equal to 0. Values may be null.
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Item Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the value associated with the specified key.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the value to get or set.

Field Value

The value associated with the specified key. If the specified key does not exist during a get operation, a KeyNotFoundException is thrown.

Implements

IDictionary TKey, TValue Item TKey
Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.
System.Collections.Generic KeyNotFoundException	Thrown when key is not found during a get operation.

Remarks

A set operation will add or replace a value in the collection, with new values being appended to the end of the collection. A key cannot be null; however a value can be. The key cannot be an int; if it is an int, then the Item Int32 property is used.
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Item Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets a ICollection T object containing all the keys in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

A ICollection T object containing all the keys in the OrderedDictionary TKey, TValue .

Implements

IDictionary TKey, TValue Keys

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets an object that can be used to synchronize this Collection.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Field Value

An object that can be used to synchronize this Collection.

Implements

ICollection SyncRoot

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Gets a ICollection T object containing all the values in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

A ICollection T object containing all the values in the OrderedDictionary TKey, TValue .

Implements

IDictionary TKey, TValue Values
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The enumerator for interating through the OrderedDictionary TKey, TValue .

Inheritance Hierarchy

System Object
  IBM.WebSphere.Caching OrderedDictionary TKey, TValue Enumerator

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A generic version of the non-generic OrderedDictionary class.

The OrderedDictionary TKey, TValue Enumerator generic type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
MoveNext	Moves the iterator to the next item in the OrderedDictionary TKey, TValue .
Reset	Moves the iterator to the first item in OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
Current	Gets the item at the iterator's current position.

Back to Top

See Also

OrderedDictionary TKey, TValue Enumerator Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue Enumerator generic type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
MoveNext	Moves the iterator to the next item in the OrderedDictionary TKey, TValue .
Reset	Moves the iterator to the first item in OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

OrderedDictionary TKey, TValue Enumerator Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Moves the iterator to the next item in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

true if the enumerator was successfully able to move the next item in the OrderedDictionary TKey, TValue ; otherwise, false is returned to indicate that the enumerator is at the end of the OrderedDictionary TKey, TValue .

Implements

IEnumerator MoveNext

See Also

OrderedDictionary TKey, TValue Enumerator Class
OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Moves the iterator to the first item in OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Implements

IEnumerator Reset

See Also

OrderedDictionary TKey, TValue Enumerator Class
OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue Enumerator generic type exposes the following members.

Properties

Name	Description
Current	Gets the item at the iterator's current position.

Back to Top

See Also

OrderedDictionary TKey, TValue Enumerator Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the item at the iterator's current position.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Implements

IEnumerator T Current

See Also

OrderedDictionary TKey, TValue Enumerator Class
OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ReplicationVotedToRollbackTransactionException exception occurs when a transaction was rolled back because some or all of the synchronous replicas did not apply the transaction.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionCallbackException
        IBM.WebSphere.Caching ReplicationVotedToRollbackTransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ReplicationVotedToRollbackTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReplicationVotedToRollbackTransactionException type exposes the following members.

Constructors

Name	Description
ReplicationVotedToRollbackTransactionException	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class.
ReplicationVotedToRollbackTransactionException(Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with a specified exception cause.
ReplicationVotedToRollbackTransactionException(String)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message.
ReplicationVotedToRollbackTransactionException(String, Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ReplicationVotedToRollbackTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ReplicationVotedToRollbackTransactionException	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class.
ReplicationVotedToRollbackTransactionException(Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with a specified exception cause.
ReplicationVotedToRollbackTransactionException(String)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message.
ReplicationVotedToRollbackTransactionException(String, Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message and exception cause.

Back to Top

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
The exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReplicationVotedToRollbackTransactionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ReplicationVotedToRollbackTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReplicationVotedToRollbackTransactionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ReplicationVotedToRollbackTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionAlreadyActiveException exception occurs to indicate that a transaction is already active for the current Session. This exception does not cause the current active transaction to be rolled back, so the isTransactionActive method returns true.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionException
        IBM.WebSphere.Caching TransactionAlreadyActiveException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionAlreadyActiveException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Constructors

Name	Description
TransactionAlreadyActiveException	Initializes a new instance of the TransactionAlreadyActiveException class.
TransactionAlreadyActiveException(Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified exception cause.
TransactionAlreadyActiveException(String)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message.
TransactionAlreadyActiveException(String, Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

Fields

Name	Description
ivTransactionRolledBa	Indicates whether the transaction was rolled back or

ck	not. (Inherited from TransactionException.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionAlreadyActiveException	Initializes a new instance of the TransactionAlreadyActiveException class.
TransactionAlreadyActiveException(Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified exception cause.
TransactionAlreadyActiveException(String)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message.
TransactionAlreadyActiveException(String, Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message and exception cause.

Back to Top

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not. (Inherited from TransactionException.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionCallbackException exception occurs when a TransactionCallback method call fails.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionCallbackException
        IBM.WebSphere.Caching ClientServerTransactionCallbackException
        IBM.WebSphere.Caching ReplicationVotedToRollbackTransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionCallbackException type exposes the following members.

Constructors

Name	Description
TransactionCallbackException	Initializes a new instance of the TransactionCallbackException class.
TransactionCallbackException(Exception)	Initializes a new instance of the TransactionCallbackException class with a specified exception cause.
TransactionCallbackException(String)	Initializes a new instance of the TransactionCallbackException class with the specified error message.
TransactionCallbackException(String, Exception)	Initializes a new instance of the TransactionCallbackException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

TransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionCallbackException	Initializes a new instance of the TransactionCallbackException class.
TransactionCallbackException(Exception)	Initializes a new instance of the TransactionCallbackException class with a specified exception cause.
TransactionCallbackException(String)	Initializes a new instance of the TransactionCallbackException class with the specified error message.
TransactionCallbackException(String, Exception)	Initializes a new instance of the TransactionCallbackException class with the specified error message and exception cause.

Back to Top

See Also

TransactionCallbackException Class
TransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionCallbackException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionCallbackException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionException exception is a general locking exception that indicates something went wrong with a transaction. Use the isTransactionActive() and wasTransactionRolledBack() methods to determine whether transaction is still active or was rolled back as a result of this exception.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionException
        IBM.WebSphere.Caching NoActiveTransactionException
        IBM.WebSphere.Caching TransactionAlreadyActiveException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Constructors

Name	Description
TransactionException(Exception, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Boolean)	Initializes a new instance of the TransactionException class with the specified error message and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(TransactionException, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, TransactionException, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Exception, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back.

Back to Top

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not.

Back to Top

Properties

	Name	Description
	Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionException(Exception, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Boolean)	Initializes a new instance of the TransactionException class with the specified error message and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(TransactionException, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, TransactionException, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Exception, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Back to Top

See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with the specified error message and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

rolledBack
: Type: System Boolean
Specifies if the trasnsaction was rolled back. A value of true indicates that the transaction was rolled back.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

TransactionException Constructor (TransactionException, Boolean)

IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: IBM.WebSphere.Caching TransactionException
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
: Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

Remarks

The cause and a detailed message of (cause==null ? null : cause.toString()) is used, which typically contains the class and detailed message of cause. This constructor is useful for as a wrapper for other Throwable objects that occur.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: IBM.WebSphere.Caching TransactionException
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
: Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

Remarks

The detailed error message that is associated with the cause is not automatically incorporated in this the detailed message for this TransactionException exception.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
: Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not.

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Indicates whether the transaction was rolled back or not.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back.

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

true if transaction is active, false if transaction never started or was complete.
See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns true if the transaction was rolled back.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Returns true if transaction was rolled back, false otherwise.

See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionTimeoutException exception occurs when a transaction exceeds the transaction timeout value that was specified on the ObjectGrid or Session.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching TransactionTimeoutException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionTimeoutException type exposes the following members.

Constructors

Name	Description
TransactionTimeoutException(String, Exception)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and exception cause.
TransactionTimeoutException(String, String)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and the transaction that timed out.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
getTxIDString	Gets the String representation of the TxID.toString() method for the transaction that timed out.
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
whenOccurred	Returns the time when this TransactionTimeoutException exception was created.

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerExc	Gets the Exception instance that caused the current exception.

	eption	(Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

TransactionTimeoutException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionTimeoutException(String, Exception)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and exception cause.
TransactionTimeoutException(String, String)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and the transaction that timed out.

Back to Top

See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionTimeoutException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
TransactionTimeoutException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionTimeoutException class with the specified error message and the transaction that timed out.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

txIdString
Type: System String
Specifies the result of the TxID.toString() method for the transaction that timed out.

See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
TransactionTimeoutException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionTimeoutException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
getTxIDString	Gets the String representation of the TxID.toString() method for the transaction that timed out.
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
whenOccurred	Returns the time when this TransactionTimeoutException exception was created.

Back to Top

See Also

TransactionTimeoutException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the String representation of the TxID.toString() method for the transaction that timed out.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the result of TxID.toString() method for the transaction that timed out.
See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns the time when this TransactionTimeoutException exception was created.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies a date object that represents the exact time when this exception object was created.
See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionTimeoutException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionTimeoutException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies an enumeration that defines the valid transaction isolation level values.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Members

Member name	Value	Description
NotSet	-1	Specifies that the transaction isolation level value is not set.
ReadUncommitted	1	Specifies that dirty reads, non-repeatable, reads and phantom reads can occur.
ReadCommitted	2	Specifies that dirty reads are prevented, but non-repeatable reads and phantom reads can occur.
RepeatableRead	4	Specifies that dirty reads and non-repeatable reads are prevented, but phantom reads can occur.

See Also

IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IBM.WebSphere.Caching.Map namespace includes the data access application programming interfaces. See the IBM.WebSphere.Caching namespace documentation for a description on how to access a map.

The eXtreme Scale client supports transactional data access to individual maps using automatic and manual transactions. The following maps are available:

- The IGridMapPessimisticAutoTx TKey, TValue interface provides automatic transactions.
- The IGridMapPessimisticTx TKey, TValue interface provides manual transaction demarcation.

See each respective interface for programming examples.

Classes

Class	Description
CacheKeyNotFoundException	A CacheKeyNotFoundException exception occurs if a key cannot be found in the cache.
ClassAliasAttribute	Specifies the class alias that you can use to correlate different class names that have the same class alias. The class alias and class fields or types are used to identify a unique class type ID during the object class serialization and de-serialization. The [ClassAlias] annotation can be specified for a user defined class.
ClientServerLoaderException	A ClientServerLoaderException exception is a base exception for any client/server operation exceptions.
DuplicateKeyException	A DuplicateKeyException exception occurs if a key cannot be inserted into the backing map because an object with the same key already exists.
FieldAliasAttribute	Specifies the field alias that you can use to correlate different class field names that have the same class field alias. The class alias and field alias or types are used to identify a unique class type ID during the object class serialization and de-serialization. The [FieldAlias] annotation can be specified for user defined class fields.
LoaderException	A LoaderException exception is the base exception that results for any exceptions that are encountered by a Loader.

[LockDeadlockException](on)	A LockStrategyNotSupportedException exception occurs when the lock manager detects a deadlock. This exception occurs to prevent the deadlock.
[LockException](on)	A LockException exception indicates errors with locking operations.
[LockStrategyNotSupportedException](on)	A LockStrategyNotSupportedException exception occurs if a map is configured with an unsupported lock strategy.
[LockTimeoutException](on)	A LockTimeoutException exception occurs when the lock manager detects that the lock wait time exceeded the maximum wait time. The timeout might be the result of a deadlock. If a deadlock is causing the timeout, the timeout is used to break the deadlock.
[MultiplePartitionWriteException](on)	A MultiplePartitionWriteException exception is a base exception for client/server operations when a user attempts to write to multiple remote partitions on remote servers in the same transaction.
[OptimisticCollisionException](on)	An OptimisticCollisionException occurs when an optimistic locking strategy is used and more than one update transaction collides on the same map entry of an ObjectGrid instance. The first transaction to commit updates the version object for the map entry. Other transactions that read this same map entry before committing have the previous version object. When the other transactions try to commit, the version object that is read does not match the version that was last committed. Therefore, other transactions are prevented from updating a map entry with stale data. The default OptimisticCallback plug-in is used by the run time if an implementation is not provided by the application. If a well-constructed equals(Object) method is not on your value object, this exception occurs because the entire value object is used as the version object.
[StaleMapEntryException](on)	Because this exception indicates that the map entry contains stale data, stale map entries or entries as identified by the key parameter that is passed to the OptimisticCollisionException(String, String, String, Object) method are invalidated. If this exception is thrown by a Loader plug-in and a null reference is used as the key parameter by the loader, the run time assumes that the loader does not know which entry caused the exception. In this scenario, the LogSequence object is passed to the Loader.batchUpdate(TxID, LogSequence) method to determine which map entries to invalidate. Each LogElement entry in the LogSequence object that is type update or

	delete is invalidated.
PartitionKeyAttribute	Specifies one or more attributes to use to calculate the partition hash code. The PartitionKey attribute can be specified for the class using a path syntax to identify a single attribute. Multiple attributes can be specified using additional PartitionKey annotations on each field, using the Order attribute to specify the order in which the hash codes will be calculated. The PartitionKey attribute is not inheritable.
ReadOnlyException	A ReadOnlyException exception occurs when a modify operation is attempted on a read-only map.
TargetNotAvailableException	A TargetNotAvailableException occurs when a remote target was not found or was not reachable.
TransactionAffinityException	A TransactionAffinityException exception occurs during server failover for inflight transactions. Applications can try the transaction again.
UnavailableServiceException	An UnavailableServiceException exception occurs when all servers are not running, or when all services are not available even though servers are running.
UndefinedMapException	An UndefinedMapException exception occurs to indicate that the map that an application tried to access is not defined in the ObjectGrid.

Interfaces

Interface	Description
IGridMap TKey, TValue	The top level interface for all maps. Use the interface to retrieve an appropriate IGridMap instance. Different IGridMap implementations are returned which allow additional operations for specific configurations and usage patterns.
IGridMapPessimisticAutoTx TKey, TValue	This is a handle to a map using automatic transaction demarcation. An instance of this IGridMapPessimisticAutoTx can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance. This is a handle to a map using the pessimistic locking strategy and

	manual transaction demarcation. All data access operations must occur within an active transaction.
IGridMapPessimisticTx TKey, TValue	Use the Transaction property to access the IGridTransaction instance that is associated with this map instance, and use the Begin method to begin a transaction. All keys in a single transaction must resolve to the same partition. An instance of this IGridMapPessimisticTx is not thread-safe, and can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance.
IPartitionManager TKey, TValue	An IPartitionManager provides properties and methods for determining how partitions are calculated. Retrieve an IPartitionManager from an IGridMap TKey, TValue .

Enumerations

Enumeration	Description
LockMode	Specifies the strength of a lock to acquire.
TtlType	Every grid map has an optional, built in timed based evictor that is referred to as the "time to live" evictor or TTL evictor. Each grid map entry has an expiration time that determines how long the entry is allowed to live in the grid map. When the expiration time is reached, the TTL evictor causes the expired entry to be evicted from the grid map. This enum defines the TTLType value constants that determine how the the expiration time is computed for a map entry.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A CacheKeyNotFoundException exception occurs if a key cannot be found in the cache.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map CacheKeyNotFoundException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

CacheKeyNotFoundException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CacheKeyNotFoundException type exposes the following members.

Constructors

Name	Description
CacheKeyNotFoundException	Initializes a new instance of the CacheKeyNotFoundException class.
CacheKeyNotFoundException(Exception)	Initializes a new instance of the CacheKeyNotFoundException class with a specified exception cause.
CacheKeyNotFoundException(String)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message.
CacheKeyNotFoundException(String, Exception)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message and exception cause.

Back to Top

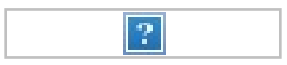
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

CacheKeyNotFoundException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
CacheKeyNotFoundException	Initializes a new instance of the CacheKeyNotFoundException class.
CacheKeyNotFoundException(Exception)	Initializes a new instance of the CacheKeyNotFoundException class with a specified exception cause.
CacheKeyNotFoundException(String)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message.
CacheKeyNotFoundException(String, Exception)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message and exception cause.

Back to Top

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CacheKeyNotFoundException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

CacheKeyNotFoundException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CacheKeyNotFoundException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

CacheKeyNotFoundException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the class alias that you can use to correlate different class names that have the same class alias. The class alias and class fields or types are used to identify a unique class type ID during the object class serialization and de-serialization.

The [ClassAlias] annotation can be specified for a user defined class.

Inheritance Hierarchy

System Object
  System Attribute
    IBM.WebSphere.Caching.Map ClassAliasAttribute

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

The syntax for the class could look like :

[ClassAlias("ACME_Employee")]

In the above example, ACME_Employee is the class alias for this user defined class.

If [ClassAlias] annotation is not defined, the name of this class is set as the ClassAlias.

Copy to ClipboardPrint

```
[ClassAlias("ACME_Employee")]
class Employee1 {
    [FieldAlias("Employee ID")]
    int empId = -1;

    [FieldAlias("Department No.")]
    int deptId = -1;

    [FieldAlias("Year Salary")]
    float salary = 0;

    [FieldAlias("SEX")]
    String sex = "M";

    int age = -1;
    String homeAddress = "";
```

When a ClassAlias and/or FieldAlias are specified in a user defined class, the ClassAlias and/or FieldAlias will be used to create or correlate with an object that are stored or will be stored in the grid. If two user defined classes (in a separate .NET application environment) have the different class name, but they were marked as the same ClassAlias, and all fields and field types are matched between these 2 classes, they will be correlated with the same class type ID even though they have the different class name. This way will allow the same class metadata to be reused between these 2 classes when running serialization and de-serialization in the different .NET application runtime, as well as to shared with Java when

the Alias for the class defined in Java and fields are also matched.

See Also

ClassAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClassAliasAttribute type exposes the following members.

Constructors

Name	Description
ClassAliasAttribute	The class alias.

Back to Top

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
Alias	The class alias.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

ClassAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The class alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

value
Type: System String
The class alias value.

See Also

ClassAliasAttribute Class
ClassAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClassAliasAttribute type exposes the following members.

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

ClassAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClassAliasAttribute type exposes the following members.

Properties

	Name	Description
	Alias	The class alias.
	TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

ClassAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The class alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

Field Value

The class alias.
See Also

ClassAliasAttribute Class
ClassAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ClientServerLoaderException exception is a base exception for any client/server operation exceptions.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LoaderException
        IBM.WebSphere.Caching.Map ClientServerLoaderException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ClientServerLoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerLoaderException type exposes the following members.

Constructors

Name	Description
ClientServerLoaderException	Initializes a new instance of the ClientServerLoaderException class.
ClientServerLoaderException(Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified exception cause.
ClientServerLoaderException(String)	Initializes a new instance of the ClientServerLoaderException class with the specified error message.
ClientServerLoaderException(String, Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ClientServerLoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ClientServerLoaderException	Initializes a new instance of the ClientServerLoaderException class.
ClientServerLoaderException(Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified exception cause.
ClientServerLoaderException(String)	Initializes a new instance of the ClientServerLoaderException class with the specified error message.
ClientServerLoaderException(String, Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified error message and exception cause.

Back to Top

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerLoaderException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerLoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerLoaderException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerLoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A DuplicateKeyException exception occurs if a key cannot be inserted into the backing map because an object with the same key already exists.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map DuplicateKeyException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

DuplicateKeyException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The DuplicateKeyException type exposes the following members.

Constructors

Name	Description
DuplicateKeyException	Initializes a new instance of the DuplicateKeyException class.
DuplicateKeyException(Exception)	Initializes a new instance of the DuplicateKeyException class with a specified exception cause.
DuplicateKeyException(String)	Initializes a new instance of the DuplicateKeyException class with the specified error message.
DuplicateKeyException(String, Exception)	Initializes a new instance of the DuplicateKeyException class with the specified error message and exception cause.

Back to Top

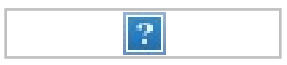
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception.

		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

DuplicateKeyException Class

IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
DuplicateKeyException	Initializes a new instance of the DuplicateKeyException class.
DuplicateKeyException(Exception)	Initializes a new instance of the DuplicateKeyException class with a specified exception cause.
DuplicateKeyException(String)	Initializes a new instance of the DuplicateKeyException class with the specified error message.
DuplicateKeyException(String, Exception)	Initializes a new instance of the DuplicateKeyException class with the specified error message and exception cause.

Back to Top

See Also

DuplicateKeyException Class
DuplicateKeyException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies an error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The DuplicateKeyException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

DuplicateKeyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The DuplicateKeyException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

DuplicateKeyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the field alias that you can use to correlate different class field names that have the same class field alias. The class alias and field alias or types are used to identify a unique class type ID during the object class serialization and de-serialization.

The [FieldAlias] annotation can be specified for user defined class fields.

Inheritance Hierarchy

System Object
  System Attribute
    IBM.WebSphere.Caching.Map FieldAliasAttribute

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

The syntax for the field alias could look as follows :

[FieldAlias("Employee ID")]

In the above example, "Employee ID" is a field alias for the empId field in this user defined class.

If [FieldAlias] annotation is not defined, the name of this field is set as the FieldAlias.

Copy to ClipboardPrint

```
[ClassAlias("ACME_Employee")]
class Employee1 {
    [FieldAlias("Employee ID")]
    int empId = -1;

    [FieldAlias("Department No.")]
    int deptId = -1;

    [FieldAlias("Year Salary")]
    float salary = 0;

    [FieldAlias("SEX")]
    String sex = "M";

    int age = -1;
    String homeAddress = "";
```

When a ClassAlias and/or FieldAlias are specified in a user defined class, the ClassAlias and/or FieldAlias will be used to create or correlate with an object that are stored or will be stored in the grid. If two user defined classes (in a separate .NET application environment) have the different class name, but they were marked as the same ClassAlias, and all fields and field types are matched between these 2 classes, they will be correlated with the same class type ID even though they have the different class name. This way will allow the same class metadata to be reused between these 2 classes when running serialization and de-

serialization in the different .NET application runtime, as well as to shared with Java when the Alias for the class defined in Java and fields are also matched.

See Also

FieldAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The FieldAliasAttribute type exposes the following members.

Constructors

Name	Description
FieldAliasAttribute	The field alias.

Back to Top

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
Alias	The field alias.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

FieldAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The field alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

value
Type: System String
The field alias value.

See Also

FieldAliasAttribute Class
FieldAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The FieldAliasAttribute type exposes the following members.

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

FieldAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The FieldAliasAttribute type exposes the following members.

Properties

	Name	Description
	Alias	The field alias.
	TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

FieldAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The field alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The field alias.

See Also

FieldAliasAttribute Class
FieldAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The top level interface for all maps. Use the interface to retrieve an appropriate IGridMap instance.

Different IGridMap implementations are returned which allow additional operations for specific configurations and usage patterns.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
: Generic type key.

TValue
: Generic type value.

See Also

IGridMap TKey, TValue _Members
IBM.WebSphere.Caching.Map Namespace
IBM.WebSphere.Caching IGrid

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMap TKey, TValue type exposes the following members.

Properties

	Name	Description
	Grid	Retrieves the IGrid instance associated with this map.
	Name	Retrieves the map name.
	PartitionManager	Retrieves the IPartitionManager associated with this map.

Back to Top

See Also

IGridMap TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMap TKey, TValue type exposes the following members.

Properties

	Name	Description
	Grid	Retrieves the IGrid instance associated with this map.
	Name	Retrieves the map name.
	PartitionManager	Retrieves the IPartitionManager associated with this map.

Back to Top

See Also

IGridMap TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the IGrid instance associated with this map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The IGrid instance.

See Also

IGridMap TKey, TValue Interface
IGridMap TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the map name.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

Field Value

The map name.
See Also

IGridMap TKey, TValue Interface
IGridMap TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the IPartitionManager associated with this map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The IPartitionManager associated with this map.

See Also

IGridMap TKey, TValue Interface
IGridMap TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

This is a handle to a map using automatic transaction demarcation.

An instance of this IGridMapPessimisticAutoTx can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
: Generic type key.

TValue
: Generic type value.

Remarks

Each data access method includes a "Specification details" table that includes the following information:

Required permission:	The permission required to use the API.
Pessimistic read lock acquired:	The type of lock that is acquired when using pessimistic locking with repeatable read transaction isolation.
Cache tier:	Identifies the map cache tiers that are included when fetching or updating cache entries in the call and under what circumstances. The following tiers are available for IGridMapPessimisticAutoTx maps: • Server Cache (data grid) • Loader (if enabled)

Examples

This sample demonstrates how to put a new cache entry into the data grid:
Copy to ClipboardPrint

```
// Assume we have already connected to the Grid...
IGrid grid = ...

// Retrieve a new map instance.
IGridMapPessimisticAutoTx<long, string> map = grid.GetGridMapPessimisticAutoTx<long, string>("MyP
essimisticMap");

try
{
    // Put the entry in the cache.
    map.Put(123, "Value to cache");
```

```
}
catch(GridException)
{
    // Handle any concequences of failed put.
}

// Dispose the map (optional, but it improves performance)
map.Dispose();
```

See Also

IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticAutoTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered, which may be during a flush or commit operation, in which the exception will be an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a nullkey cannot be specified.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values associated with the list of keys that are specified in the keyList. If the value is not found, a null is returned. If the map supports null values, use the ContainsKeyAll(IList TKey) to test for multiple keys that may have a null value.
Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding a new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method

	in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException is thrown during commit.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException is thrown.
TouchAll	Updates the last access time for the data grid entries specified in the keyList without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
TryAdd	A variant of Add(TKey, TValue) that does not throw exceptions.
TryAddAll	Adds the key-value pairs to the data grid.
TryInvalidate	Removes the entry associated with the specified key from the cache, leaving the data behind unchanged.
TryInvalidateAll	Removes the entries associated with the keys that are specified in the keyList from the cache, leaving the data behind unchanged.
TryPut	Adds the key-value pair to the grid. If an entry for the key exists in the data grid, the value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryPutAll	Adds each key-value pair to the data grid. If an entry for a key exists in the data grid, its value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryRemove	Removes the entry that is associated with the specified key from the data grid. If the key has no matching entry in the map, no action is taken.
TryRemoveAll	Removes the list of entries associated with the keys that are specified in the keyList. If a key in the keyList has no matching entry in the map, no action is taken for that key.
TryReplace	Replaces the value of a key-value pair in the data grid. If an entry for the key exists in the data grid, its value is replaced with the specified value.

TryReplaceAll	Replaces each key-value pair that is specified in the dictionary object in the data grid. If an entry for the key exists in the data grid, its value is replaced with the corresponding value specified.
TryTouch	Updates the last access time for the data grid entry that matches the key.
TryTouchAll	Updates the last access time for the data grid entries that match each key in the keyList.

Back to Top

Properties

Name	Description
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	Gets or sets the value in the map using the specified key.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
MapName	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
PartitionManager	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever".

Live	This property can only be used when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException is thrown. To disable the TTL timeout, use a value of TimeSpan.Zero. To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.
TtlEvictorType	Gets the TtlType of the map's TTL evictor.

Back to Top
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticAutoTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered, which may be during a flush or commit operation, in which the exception will be an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a nullkey cannot be specified.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values associated with the list of keys that are specified in the keyList. If the value is not found, a null is returned. If the map supports null values, use the ContainsKeyAll(IList TKey) to test for multiple keys that may have a null value.
Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding a new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method

	in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException is thrown during commit.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException is thrown.
TouchAll	Updates the last access time for the data grid entries specified in the keyList without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
TryAdd	A variant of Add(TKey, TValue) that does not throw exceptions.
TryAddAll	Adds the key-value pairs to the data grid.
TryInvalidate	Removes the entry associated with the specified key from the cache, leaving the data behind unchanged.
TryInvalidateAll	Removes the entries associated with the keys that are specified in the keyList from the cache, leaving the data behind unchanged.
TryPut	Adds the key-value pair to the grid. If an entry for the key exists in the data grid, the value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryPutAll	Adds each key-value pair to the data grid. If an entry for a key exists in the data grid, its value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryRemove	Removes the entry that is associated with the specified key from the data grid. If the key has no matching entry in the map, no action is taken.
TryRemoveAll	Removes the list of entries associated with the keys that are specified in the keyList. If a key in the keyList has no matching entry in the map, no action is taken for that key.
TryReplace	Replaces the value of a key-value pair in the data grid. If an entry for the key exists in the data grid, its value is replaced with the specified value.

TryReplaceAll	Replaces each key-value pair that is specified in the dictionary object in the data grid. If an entry for the key exists in the data grid, its value is replaced with the corresponding value specified.
TryTouch	Updates the last access time for the data grid entry that matches the key.
TryTouchAll	Updates the last access time for the data grid entries that match each key in the keyList.

Back to Top

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface

IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pair to the data grid.

The key must not exist before executing this method.

A DuplicateKeyException is thrown when the duplicate key is discovered.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be inserted in the data grid.

value
Type: TValue
Specifies the value to be inserted in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when the key exists.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INSERT
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds multiple key-value pairs to the data grid.

The keys must not exist before executing this method.

A DuplicateKeyException is thrown when the duplicate key is discovered, which may be during a flush or commit operation, in which the exception will be an inner exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to be inserted into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullentries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when a dictionary key in entries already exists in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission: MapPermission.INSERT	
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a nullkey cannot be specified.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to test in the map.

Return Value

Returns true if the key is found, false otherwise.
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys to test in the map.

Return Value

Specifies a list of boolean values. If the key is found in the keyList, true is listed. Otherwise, false is returned.
Exceptions

Exception	Condition
System ArgumentException	Occurs when keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the value that is associated with the specified key. If the value is not found, a null is returned.

If the map supports null values, use the ContainsKey(TKey) to test for a key that may have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to query.

Return Value

The value that is associated with the specified key if it exists; otherwise null is returned.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified, or when the data type of the value that was obtained from the data grid does not match the declared data type.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the values associated with the list of keys that are specified in the keyList. If the value is not found, a null is returned.

If the map supports null values, use the ContainsKeyAll(IList TKey) to test for multiple keys that may have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to query.

Return Value

A list of values that are associated with the supplied keys. If the value associated with a particular key is not in the data grid, null is returned in the list at the position that is associated with the key.
Exceptions

Exception	Condition
System ArgumentException	Occurs when the keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing,
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store).

If the key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store).

If a key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be put in the data grid.

value
Type: TValue
Specifies the value to be put in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts multiple key-value pairs to the data grid, replacing or adding a new entries to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to be put into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullentries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store).

If the key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store).

If a key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store).

If the key cannot be found in the data grid a CacheKeyNotFoundException is thrown during commit.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be updated.

value
Type: TValue
Specifies the value to be updated in the data grid and Loader.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store).

If a key cannot be found in the map a CacheKeyNotFoundException is thrown.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to replace in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within entries.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs when the key is not found in any of the cache tiers or Loader. This may be deferred until commit time.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Resets the configurable settings for the map back to configured values.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

This method only resets configuration parameters that can be overridden by the client.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value.

If the key cannot be found in the map a CacheKeyNotFoundException is thrown.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs if the key does not exist in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Updates the last access time for the data grid entries specified in the keyList without locking the entries or fetching the values.

If a key cannot be found in the map a CacheKeyNotFoundException is thrown.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the keys to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs if the key does not exist in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A variant of Add(TKey, TValue) that does not throw exceptions.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be inserted in the data grid.

value
Type: TValue
Specifies the value to be inserted in the data grid.

Return Value

Returns true if successful, false if unsuccessful.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pairs to the data grid.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to be inserted into the data grid.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If entries is null or a null key is specified in a dictionary entry or a dictionary key in entries exists, false is returned. If all keys in the dictionary entries do not resolve to the same partition or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Removes the entry associated with the specified key from the cache, leaving the data behind unchanged.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for which you want to remove the value.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If key is null or the caller does not have authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Removes the entries associated with the keys that are specified in the keyList from the cache, leaving the data behind unchanged.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
The list of keys whose values are to be removed.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If keyList is null or a null key is specified in a keyList entry, false is returned. If all keys in the keyList do not resolve to the same partition or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pair to the grid. If an entry for the key exists in the data grid, the value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
: Type: TKey
specifies the key to be inserted into the data grid.

value
: Type: TValue
specifies the value to be inserted into the data grid.

Return Value

Returns true if successful, false if unsuccessful.

Remarks

If the key is null, or the caller has insufficient authority to run this operation, false is returned.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds each key-value pair to the data grid. If an entry for a key exists in the data grid, its value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object that contains the key-value pairs to be added to the data grid.

Return Value

Returns true if successful, false if unsuccessful.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid. If the key has no matching entry in the map, no action is taken.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for which you want to remove the entry.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If key is null or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Removes the list of entries associated with the keys that are specified in the keyList. If a key in the keyList has no matching entry in the map, no action is taken for that key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
The list of keys whose entries are to be removed.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If keyList is null or a null key is specified in a keyList entry, false is returned. If all keys in the keyList do not resolve to the same partition or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Replaces the value of a key-value pair in the data grid. If an entry for the key exists in the data grid, its value is replaced with the specified value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for the entry to be replaced.

value
Type: TValue
Specifies the new value.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If key is null or the key does not match an existing entry in the data grid, false is returned. If the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Replaces each key-value pair that is specified in the dictionary object in the data grid. If an entry for the key exists in the data grid, its value is replaced with the corresponding value specified.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object that contains key-value pairs to be replaced in the data grid.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If entries is null, a null key is specified in a dictionary entry, or the key does not match an existing entry in the data grid, false is returned. If all keys in the dictionary entries do not resolve to the same partition, or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Updates the last access time for the data grid entry that matches the key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to have last access time updated.

Return Value

Returns true if successful, false if unsuccessful.

Remarks

If the key is null or the key does not match an existing entry in the data grid, false is returned.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Updates the last access time for the data grid entries that match each key in the keyList.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys to have last access time updated.

Return Value

Returns true if successful, false if unsuccessful.

Remarks

If an entry is null or any key in the keyList is null, false is returned. If a key in the keyList does not match an existing entry in the data grid or all keys in the keyList do not resolve to the same partition, false is returned.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticAutoTx TKey, TValue type exposes the following members.

Properties

Name	Description
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	Gets or sets the value in the map using the specified key.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
MapName	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
PartitionManager	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever". This property can only be used when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException is thrown.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

TtlEvictorType

Gets the TtlType of the map's TTL evictor.

Back to Top
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the value in the map using the specified key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the value to set or get.

Field Value

The value associated with key to get or set.

Remarks

This indexer retreives or put's the key and value into the map using the Get(TKey) and Put(TKey, TValue) methods respectively.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace
IGridMapPessimisticAutoTx TKey, TValue Get(TKey)
IGridMapPessimisticAutoTx TKey, TValue Put(TKey, TValue)

Send Feedback on this topic to WAS Documentation Team.

Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The maximum time to wait when acquiring a lock on an item in the grid map.
Exceptions

Exception	Condition
System ArgumentException	Occurs if the value is not >= Zero.

Remarks

To prevent deadlocks from occurring, the grid map has a default timeout value of 15 seconds; however, on a heavily loaded system, lock timeouts can occur without an actual deadlock. Use this property to increase the value from the default to prevent false lock timeout exceptions from occurring.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever".

This property can only be used when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException is thrown.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Exceptions

Exception	Condition
System ArgumentException	Occurs if the TimeSpan is not >=0 or TimeSpan.MinValue or if the TtlEvictorType property is not LastAccessTime or LastUpdateTime.

Remarks

Required Permission: MapPermission.INVALIDATE

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the TtlType of the map's TTL evictor.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The TtlType of the map's TTL evictor.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

This is a handle to a map using the pessimistic locking strategy and manual transaction demarcation. All data access operations must occur within an active transaction.

Use the Transaction property to access the IGridTransaction instance that is associated with this map instance, and use the Begin method to begin a transaction. All keys in a single transaction must resolve to the same partition.

An instance of this IGridMapPessimisticTx is not thread-safe, and can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
Generic type key.

TValue
Generic type value.

Remarks

Each data access method includes a "Specification details" table that includes the following information:

Required permission:	The permission required to use the API.
Pessimistic read lock acquired:	The type of lock that is acquired when you are using pessimistic locking with repeatable read transaction isolation.
Pessimistic read lock held:	The type of lock that is held for the duration of the transaction with repeatable read transaction isolation. Locks can be upgraded but not demoted during a transaction.
Cache tier:	Identifies the map cache tiers that are included when fetching or updating cache entries in the call and under what circumstances. The following tiers are available for IGridMapPessimisticTx maps: • Transactional Cache • Server Cache (data grid) • Loader (if enabled)

Examples

This sample demonstrates how to put a new cache entry into the data grid:
Copy to ClipboardPrint

```
// Assume we have already connected to the Grid...
IGrid grid = ...

// Retrieve a new map instance.
IGridMapPessimisticTx<long, string> map = grid.GetGridMapPessimisticTx<long, string>("MyPessimist
icMap");

// Start a transaction.
map.Transaction.Begin();

try
{
    // Lock the entry in the data grid with an Upgradable lock.
    map.Lock(123, LockMode.Upgradable);

    // Put the entry in the transactional cache.
    map.Put(123, "Value to cache");

    // Commit the transaction to the data grid.
    map.Transaction.Commit();
}
catch(GridException)
{
    // Clean-up the transaction if the lock could not be
    // acquired, or the commit failed.
    if(map.Transaction.Active)
    {
        try
        {
            map.Transaction.Rollback();
        }
        catch(Exception)
        {
            // Optionally log this exception, or ignore the exception.
        }
    }

    // Dispose the map (optional, but it improves performance)
    map.Dispose();

    // Rethrow the real exception.
    throw;
}
```

See Also

IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before you run this method. A DuplicateKeyException exception results when the duplicate key is discovered. Duplicate keys might be discovered with a flush or commit operation. In this scenario, the exception is an inner exception.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException exception results when the duplicate key is discovered. This discovery might happen during a flush or commit operation. In this scenario, the exception is an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a null key cannot be specified. This API does not hold any locks. Use the Lock(TKey, LockMode) to test for a key and retain a lock.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList. This API does not hold any locks. Use the LockAll(IList TKey , LockMode) to test for a key and retain a lock.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values that are associated with the list of keys that are specified in the keyList. If the value is not found, a null value is returned. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for multiple keys that might have a null value.
GetAndLock	Locks the specified key and retrieves the associated value. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that might have a null value.
GetAndLockAll	Locks the specified keys and retrieves the associated values. If a value is not found, a null is returned in the list. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for a key that might have a null value.

Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Lock	Locks the specified key and tests to see if the key was previously present in the data grid or Loader.
LockAll	Locks the specified keys and tests to see if each was previously present in the data grid or Loader.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException exception results during the commit operation.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException exception results.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.
TouchAll	Updates the last access time for the data grid entries that are specified in the keyList list without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Back to Top

Properties

Name	Description

Name	
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	An indexer that retreives or puts the key and value into the map with the Get(TKey) and Put(TKey, TValue) methods.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
Name	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
Partition	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
PartitionManager	
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever". This property can be used only when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException exception results. To disable the TTL timeout, use a value of TimeSpan.Zero. To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.
Transact	The Transaction instance used to configure and demarcate a transaction. (Inherited from ITransactionable.)

ionTtlEvictorType Retrieves the time to live type for the evictor on the map.

Back to Top
See Also

IGridMapPessimisticTx TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before you run this method. A DuplicateKeyException exception results when the duplicate key is discovered. Duplicate keys might be discovered with a flush or commit operation. In this scenario, the exception is an inner exception.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException exception results when the duplicate key is discovered. This discovery might happen during a flush or commit operation. In this scenario, the exception is an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a null key cannot be specified. This API does not hold any locks. Use the Lock(TKey, LockMode) to test for a key and retain a lock.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList. This API does not hold any locks. Use the LockAll(IList TKey , LockMode) to test for a key and retain a lock.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values that are associated with the list of keys that are specified in the keyList. If the value is not found, a null value is returned. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for multiple keys that might have a null value.
GetAndLock	Locks the specified key and retrieves the associated value. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that might have a null value.
GetAndLockAll	Locks the specified keys and retrieves the associated values. If a value is not found, a null is returned in the list. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for a key that might have a null value.

Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Lock	Locks the specified key and tests to see if the key was previously present in the data grid or Loader.
LockAll	Locks the specified keys and tests to see if each was previously present in the data grid or Loader.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException exception results during the commit operation.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException exception results.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.
TouchAll	Updates the last access time for the data grid entries that are specified in the keyList list without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Back to Top

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pair to the data grid.

The key must not exist before you run this method.

A DuplicateKeyException exception results when the duplicate key is discovered. Duplicate keys might be discovered with a flush or commit operation. In this scenario, the exception is an inner exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be inserted in the data grid.

value
Type: TValue
Specifies the value to be inserted in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullkey is specified.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when the key exists.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INSERT
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds multiple key-value pairs to the data grid.

The keys must not exist before executing this method.

A DuplicateKeyException exception results when the duplicate key is discovered. This discovery might happen during a flush or commit operation. In this scenario, the exception is an inner exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
: Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a IOrderedDictionary TKey, TValue object of key-value pairs to be inserted into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullentries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when a dictionary key in entries already exists in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INSERT
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a null key cannot be specified.

This API does not hold any locks. Use the Lock(TKey, LockMode) to test for a key and retain a lock.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to test in the map.

Return Value

Returns true if the key is found, false otherwise.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	No
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.

This API does not hold any locks. Use the LockAll(IList TKey , LockMode) to test for a key and retain a lock.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys to test in the map.

Return Value

Specifies a list of boolean values. If the key is found in the keyList, true is listed. Otherwise, false is returned.
Exceptions

Exception	Condition
System ArgumentException	Occurs when keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	No
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the value that is associated with the specified key. If the value is not found, a null is returned.

If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that may have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to query.

Return Value

The value that is associated with the specified key if it exists; otherwise null is returned.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified, or when the data type of the value that was obtained from the data grid does not match the declared data type.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the values that are associated with the list of keys that are specified in the keyList. If the value is not found, a null value is returned.

If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for multiple keys that might have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to query.

Return Value

A list of values that are associated with the supplied keys. If the value associated with a particular key is not in the data grid, null is returned in the list at the position that is associated with the key.

Exceptions

Exception	Condition
System ArgumentException	Occurs when the keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing,
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	Yes
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Locks the specified key and retrieves the associated value. If the value is not found, a null is returned.

If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that might have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

Specifies the value that is associated with the specified key if it exists. Otherwise, null is returned.
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Locks the specified keys and retrieves the associated values. If a value is not found, a null is returned in the list.

If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for a key that might have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

A list of values that are associated with the supplied keys. If the value associated with a particular key is not in the data grid, null is returned in the list at the position that is associated with the key.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the keys are found and the appropriate locks are acquired.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store).

If the key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store).

If a key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Locks the specified key and tests to see if the key was previously present in the data grid or Loader.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

Returns true if the key is found in the data grid or Loader (back-end persistent store).
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Locks the specified keys and tests to see if each was previously present in the data grid or Loader.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

A list of bool that are associated with the supplied keys, where true indicates that the key was found in the data grid or Loader (back-end persistent store).
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
: Type: TKey
Specifies the key to be put in the data grid.

value
: Type: TValue
Specifies the value to be put in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts multiple key-value pairs to the data grid, replacing or adding new entries to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a IOrderedDictionary TKey, TValue object of key-value pairs to be put into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null entries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store).

If the key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store).

If a key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store).

If the key cannot be found in the data grid a CacheKeyNotFoundException exception results during the commit operation.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be updated.

value
Type: TValue
Specifies the value to be updated in the data grid and Loader.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store).

If a key cannot be found in the map a CacheKeyNotFoundException exception results.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a IOrderedDictionary TKey, TValue object of key-value pairs to replace in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within entries.
IBM.WebSphere.Caching Grid Exception	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs when the key is not found in any of the cache tiers or Loader. This exception might be deferred until commit time.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Resets the configurable settings for the map back to configured values.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Remarks

This method resets configuration parameters that can be overridden by the client only.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value.

If the key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccesControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Updates the last access time for the data grid entries that are specified in the keyList list without locking the entries or fetching the values.

If a key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the keys to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticTx TKey, TValue type exposes the following members.

Properties

Name	Description
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	An indexer that retreives or puts the key and value into the map with the Get(TKey) and Put(TKey, TValue) methods.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
Name	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
PartitionManager	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever". This property can be used only when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException exception results.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

Transaction	The Transaction instance used to configure and demarcate a transaction. (Inherited from ITransactionable.)
TtlEvictor Type	Retrieves the time to live type for the evictor on the map.

Back to Top

See Also

IGridMapPessimisticTx TKey, TValue _Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An indexer that retreives or puts the key and value into the map with the Get(TKey) and Put(TKey, TValue) methods.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Field Value

The maximum time to wait when acquiring a lock on an item in the grid map.

Exceptions

Exception	Condition
System ArgumentException	Occurs if the value is not >= Zero.

Remarks

To prevent deadlocks from occurring, the grid map has a default timeout value of 15 seconds; however, on a heavily loaded system, lock timeouts can occur without an actual deadlock. Use this property to increase the value from the default to prevent false lock timeout exceptions from occurring.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever".

This property can be used only when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException exception results.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Exceptions

Exception	Condition
System ArgumentException	Occurs if the TimeSpan is not >=0 or TimeSpan.MinValue or if the TtlEvictorType property is not LastAccessTime or LastUpdateTime.

Remarks

Required Permission: MapPermission.INVALIDATE

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the time to live type for the evictor on the map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An IPartitionManager provides properties and methods for determining how partitions are calculated. Retrieve an IPartitionManager from an IGridMap TKey, TValue .

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
: Specifies a generic type key.

TValue
: Specifies a generic type value.

Remarks

The partition id is calculated as follows:

- The key class is examined for the PartitionKeyAttribute attribute. If present, the GetHashCode of the referenced attributes is used to calculate the partition.
- Otherwise, the key's GetHashCode method is used to calculate the partition id.

See Also

IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IPartitionManager TKey, TValue type exposes the following members.

Methods

Name	Description
GetPartition	Returns the partition number for the specified map element key.
GetPartitionAll(IList TKey)	Returns a map of partition numbers to map keys from a collection of map element keys.
GetPartitionAll(IOrderedDictionary TKey, TValue)	Returns a map of partition numbers to map keys from a collection of map element keys.

Back to Top

Properties

Name	Description
NumPartitions	Retrieves the number of partitions that are defined for the map.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IPartitionManager TKey, TValue type exposes the following members.

Methods

Name	Description
GetPartition	Returns the partition number for the specified map element key.
GetPartitionAll(IList TKey)	Returns a map of partition numbers to map keys from a collection of map element keys.
GetPartitionAll(IOrderedDictionary TKey, TValue)	Returns a map of partition numbers to map keys from a collection of map element keys.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns the partition number for the specified map element key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for which to return a partition number.

Return Value

Returns the partition number for the specified map element key.

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GetPartitionAll(IList TKey)	Returns a map of partition numbers to map keys from a collection of map element keys.
GetPartitionAll(IOrderedDictionary TKey, TValue)	Returns a map of partition numbers to map keys from a collection of map element keys.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map of partition numbers to map keys from a collection of map element keys.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys for which you want to return partition numbers.

Return Value

Returns a collection of partition numbers and associated keys.

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
GetPartitionAll Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map of partition numbers to map keys from a collection of map element keys.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

entries
: Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a dictionary object of key-value pairs for which you want to return partition numbers.

Return Value

Specifies a collection of partition numbers and associated key-value pairs.

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
GetPartitionAll Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IPartitionManager TKey, TValue type exposes the following members.

Properties

	Name	Description
	NumPartitions	Retrieves the number of partitions that are defined for the map.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the number of partitions that are defined for the map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

LoaderException Class IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LoaderException exception is the base exception that results for any exceptions that are encountered by a Loader.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LoaderException
        IBM.WebSphere.Caching.Map ClientServerLoaderException
        IBM.WebSphere.Caching.Map UnavailableServiceException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LoaderException type exposes the following members.

Constructors

Name	Description
LoaderException	Initializes a new instance of the LoaderException class.
LoaderException(Exception)	Initializes a new instance of the LoaderException class with a specified exception cause.
LoaderException(String)	Initializes a new instance of the LoaderException class with the specified error message.
LoaderException(String, Exception)	Initializes a new instance of the LoaderException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LoaderException	Initializes a new instance of the LoaderException class.
LoaderException(Exception)	Initializes a new instance of the LoaderException class with a specified exception cause.
LoaderException(String)	Initializes a new instance of the LoaderException class with the specified error message.
LoaderException(String, Exception)	Initializes a new instance of the LoaderException class with the specified error message and exception cause.

Back to Top

See Also

LoaderException Class
LoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LoaderException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LoaderException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockStrategyNotSupportedException exception occurs when the lock manager detects a deadlock. This exception occurs to prevent the deadlock.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockException
        IBM.WebSphere.Caching.Map LockTimeoutException
          IBM.WebSphere.Caching.Map LockDeadlockException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

Typically, a deadlock is the result of the following scenario: One transaction gets a map entry, resulting in a weaker lock than an existing lock on the same entry. At commit time, the transaction attempts to promote the weaker lock to a stronger lock to apply the changes to the data store. For example, two transactions try to promote from shared locks to exclusive locks, but each transaction already owns a shared lock.
See Also

LockDeadlockException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockDeadlockException type exposes the following members.

Constructors

Name	Description
LockDeadlockException	Initializes a new instance of the LockDeadlockException class.
LockDeadlockException(Exception)	Initializes a new instance of the LockDeadlockException class with the specified exception cause.
LockDeadlockException(String)	Initializes a new instance of the LockDeadlockException class with the specified error message.
LockDeadlockException(String, Exception)	Initializes a new instance of the LockDeadlockException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor. (Inherited from LockTimeoutException.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

	(Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred. (Inherited from LockTimeoutException.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockDeadlockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockDeadlockException	Initializes a new instance of the LockDeadlockException class.
LockDeadlockException(Exception)	Initializes a new instance of the LockDeadlockException class with the specified exception cause.
LockDeadlockException(String)	Initializes a new instance of the LockDeadlockException class with the specified error message.
LockDeadlockException(String, Exception)	Initializes a new instance of the LockDeadlockException class with the specified error message and exception cause.

Back to Top

See Also

LockDeadlockException Class
LockDeadlockException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockDeadlockException type exposes the following members.

Methods

	Name	Description
	Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
	Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
	GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
	getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor. (Inherited from LockTimeoutException.)
	GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LockDeadlockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockDeadlockException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred. (Inherited from LockTimeoutException.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LockDeadlockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

LockException Class IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockException exception indicates errors with locking operations.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockException
        IBM.WebSphere.Caching.Map LockTimeoutException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockException Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockException type exposes the following members.

Constructors

Name	Description
LockException	Initializes a new instance of the LockException class.
LockException(Exception)	Initializes a new instance of the LockException class with a specified exception cause.
LockException(String)	Initializes a new instance of the LockException class with the specified error message.
LockException(String, Exception)	Initializes a new instance of the LockException class with the specified error message and exception cause.

Back to Top

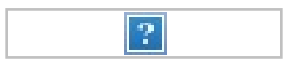
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockException	Initializes a new instance of the LockException class.
LockException(Exception)	Initializes a new instance of the LockException class with a specified exception cause.
LockException(String)	Initializes a new instance of the LockException class with the specified error message.
LockException(String, Exception)	Initializes a new instance of the LockException class with the specified error message and exception cause.

Back to Top

See Also

LockException Class
LockException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockException type exposes the following members.
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top
See Also

LockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockException type exposes the following members.
Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Specifies the strength of a lock to acquire.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Members

Member name	Value	Description
Shared	0	Shared lock that can be obtained if no one is currently holding an exclusive lock.
Upgradable	1	Upgradable lock that can be obtained if no one is currently holding an upgradable or exclusive lock.
Exclusive	2	Exclusive lock that can only be obtained if no locks are held.

See Also

IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockStrategyNotSupportedException exception occurs if a map is configured with an unsupported lock strategy.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockStrategyNotSupportedException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockStrategyNotSupportedException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockStrategyNotSupportedException type exposes the following members.

Constructors

Name	Description
LockStrategyNotSupportedException	Initializes a new instance of the LockStrategyNotSupportedException class.
LockStrategyNotSupportedException(Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with a specified exception cause.
LockStrategyNotSupportedException(String)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message.
LockStrategyNotSupportedException(String, Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockStrategyNotSupportedException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockStrategyNotSupportedException	Initializes a new instance of the LockStrategyNotSupportedException class.
LockStrategyNotSupportedException(Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with a specified exception cause.
LockStrategyNotSupportedException(String)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message.
LockStrategyNotSupportedException(String, Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message and exception cause.

Back to Top

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockStrategyNotSupportedException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LockStrategyNotSupportedException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockStrategyNotSupportedException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LockStrategyNotSupportedException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockTimeoutException exception occurs when the lock manager detects that the lock wait time exceeded the maximum wait time. The timeout might be the result of a deadlock. If a deadlock is causing the timeout, the timeout is used to break the deadlock.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockException
        IBM.WebSphere.Caching.Map LockTimeoutException
          IBM.WebSphere.Caching.Map LockDeadlockException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockTimeoutException type exposes the following members.

Constructors

Name	Description
LockTimeoutException	Initializes a new instance of the LockTimeoutException class.
LockTimeoutException(Exception)	Initializes a new instance of the LockTimeoutException class with a specified exception cause.
LockTimeoutException(String)	Initializes a new instance of the LockTimeoutException class with the specified error message.
LockTimeoutException(String, Exception)	Initializes a new instance of the LockTimeoutException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred.
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockTimeoutException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockTimeoutException	Initializes a new instance of the LockTimeoutException class.
LockTimeoutException(Exception)	Initializes a new instance of the LockTimeoutException class with a specified exception cause.
LockTimeoutException(String)	Initializes a new instance of the LockTimeoutException class with the specified error message.
LockTimeoutException(String, Exception)	Initializes a new instance of the LockTimeoutException class with the specified error message and exception cause.

Back to Top

See Also

LockTimeoutException Class
LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
: Specifies the error message that explains the reason for the exception.

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockTimeoutException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LockTimeoutException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the detailed message string of this LockTimeoutException instance.

See Also

LockTimeoutException Class
LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

The LockTimeoutException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred.
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LockTimeoutException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

Gets the value that was set in the LockRequestQueueDetails property. If the LockRequestQueueDetails property was not previously set for this exception, the return value is null.

Sets the details of the lock requests on the lock request queue at the time the lock timeout occurred.

See Also

LockTimeoutException Class
LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A MultiplePartitionWriteException exception is a base exception for client/server operations when a user attempts to write to multiple remote partitions on remote servers in the same transaction.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map MultiplePartitionWriteException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

MultiplePartitionWriteException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MultiplePartitionWriteException type exposes the following members.

Constructors

Name	Description
MultiplePartitionWriteException	Initializes a new instance of the MultiplePartitionWriteException class.
MultiplePartitionWriteException(Exception)	Initializes a new instance of the MultiplePartitionWriteException class with a specified exception cause.
MultiplePartitionWriteException(String)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message.
MultiplePartitionWriteException(String, Exception)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

MultiplePartitionWriteException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
MultiplePartitionWriteException	Initializes a new instance of the MultiplePartitionWriteException class.
MultiplePartitionWriteException(Exception)	Initializes a new instance of the MultiplePartitionWriteException class with a specified exception cause.
MultiplePartitionWriteException(String)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message.
MultiplePartitionWriteException(String, Exception)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message and exception cause.

Back to Top

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MultiplePartitionWriteException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

MultiplePartitionWriteException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MultiplePartitionWriteException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

MultiplePartitionWriteException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An OptimisticCollisionException occurs when an optimistic locking strategy is used and more than one update transaction collides on the same map entry of an ObjectGrid instance. The first transaction to commit updates the version object for the map entry. Other transactions that read this same map entry before committing have the previous version object. When the other transactions try to commit, the version object that is read does not match the version that was last committed. Therefore, other transactions are prevented from updating a map entry with stale data.

The default OptimisticCallback plug-in is used by the run time if an implementation is not provided by the application. If a well-constructed equals(Object) method is not on your value object, this exception occurs because the entire value object is used as the version object.

Because this exception indicates that the map entry contains stale data, stale map entries or entries as identified by the key parameter that is passed to the OptimisticCollisionException(String, String, String, Object) method are invalidated. If this exception is thrown by a Loader plug-in and a null reference is used as the key parameter by the loader, the run time assumes that the loader does not know which entry caused the exception. In this scenario, the LogSequence object is passed to the Loader.batchUpdate(TxID, LogSequence) method to determine which map entries to invalidate. Each LogElement entry in the LogSequence object that is type update or delete is invalidated.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map OptimisticCollisionException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OptimisticCollisionException type exposes the following members.

Constructors

Name	Description
OptimisticCollisionException	Initializes a new instance of the OptimisticCollisionException class with the specified error message, data grid name, map name, and keys that caused the exception.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
getGridName	Gets the name of the ObjectGrid instance in which the optimistic collision occurred.
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getKey	Gets key object or array of key objects that caused the OptimisticCollisionException to occur.
getMapName	Gets the map name in which the OptimisticCollisionException exception occurred.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
setKey	Set the key that caused this exception to occur.
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

OptimisticCollisionException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OptimisticCollisionException class with the specified error message, data grid name, map name, and keys that caused the exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

gridName
Type: System String
Specfies the name of the data grid to which the map belongs.

mapName
Type: System String
Specifies the name of the map.

key
Type: System Object
Specifies the key or array of keys that caused the optimistic collision exception to occur.

Remarks

If more than a single key caused the exception, use an array object for this parameter. Each array element identifies a single map entry that caused the exception to occur. Using array elements is useful when a Loader uses the batch update support of a Java Database Connectivity (JDBC) driver. Pass a null reference if you are unable to determine which key or set of keys caused this exception to occur.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OptimisticCollisionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
getGridName	Gets the name of the ObjectGrid instance in which the optimistic collision occurred.
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getKey	Gets key object or array of key objects that caused the OptimisticCollisionException to occur.
getMapName	Gets the map name in which the OptimisticCollisionException exception occurred.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
setKey	Set the key that caused this exception to occur.
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

OptimisticCollisionException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the name of the ObjectGrid instance in which the optimistic collision occurred.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the name of the ObjectGrid instance.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets key object or array of key objects that caused the OptimisticCollisionException to occur.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the key object or array of key objects that caused the exception to occur.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the map name in which the OptimisticCollisionException exception occurred.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the map name where the exception occurred.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Set the key that caused this exception to occur.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: System Object
the key or array of key objects that caused the exception

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OptimisticCollisionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

OptimisticCollisionException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies one or more attributes to use to calculate the partition hash code.

The PartitionKey attribute can be specified for the class using a path syntax to identify a single attribute. Multiple attributes can be specified using additional PartitionKey annotations on each field, using the Order attribute to specify the order in which the hash codes will be calculated.

The PartitionKey attribute is not inheritable.

Inheritance Hierarchy

System Object
  System Attribute
    IBM.WebSphere.Caching.Map PartitionKeyAttribute

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

The following examples illustrate how to identify a top-level, primitive field:

Simple class attribute example:

Copy to ClipboardPrint

```
[PartitionKey("deptId")] class Employee { int empId; int deptId; }
```

Simple field attribute example:

Copy to ClipboardPrint

```
class Employee { int empId; [PartitionKey] int deptId; }
```

Simple, multiple field attribute example:

Copy to ClipboardPrint

```
class Employee { int empId; [PartitionKey(order=0)] int deptId; [PartitionKey(order=1)] String coun
```

The following examples illustrate how to address an attribute inside an embedded class. In this case, the path separater is defined in the data grid as a "." character (the default for eXtreme Data Format):

Embedded class annotation example:

Copy to ClipboardPrint

```
[PartitionKey("deptKey.id")] class Employee { DepartmentKey deptKey; } class DepartmentKey { int id
```

Embedded field annotation example:

Copy to ClipboardPrint

```
class Employee { [PartitionKey("id")] DepartmentKey deptKey; } class DepartmentKey { int id; }
```

See Also

PartitionKeyAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The PartitionKeyAttribute type exposes the following members.

Constructors

Name	Description
PartitionKeyAttribute	An attribute-less constructor for use when defined on a field, where the field name is the attribute.
PartitionKeyAttribute(String)	Specifies a single attribute to use to calculate the partition hash code.

Back to Top

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
AttributeName	Identifies the path to the attribute that should be included as part of the partition key.
Order	Order of the fields that contribute to the partition calculation. The order values must be unique for all PartitionKey attributes defined on a key class.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

PartitionKeyAttribute Class
IBM.WebSphere.Caching.Map Namespace

information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
PartitionKeyAttribute	An attribute-less constructor for use when defined on a field, where the field name is the attribute.
PartitionKeyAttribute(String)	Specifies a single attribute to use to calculate the partition hash code.

Back to Top

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An attribute-less constructor for use when defined on a field, where the field name is the attribute.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
PartitionKeyAttribute Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies a single attribute to use to calculate the partition hash code.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

attributeName
Type: System String
The attribute name.

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
PartitionKeyAttribute Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The PartitionKeyAttribute type exposes the following members.

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

PartitionKeyAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The PartitionKeyAttribute type exposes the following members.

Properties

Name	Description
AttributeName	Identifies the path to the attribute that should be included as part of the partition key.
Order	Order of the fields that contribute to the partition calculation. The order values must be unique for all PartitionKey attributes defined on a key class.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

PartitionKeyAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Documentation d'IBM WebSphere DataPower XC10 Appliance version 2.5

Bienvenue dans la documentation d'IBM® WebSphere DataPower XC10 Appliance, dans laquelle vous trouverez des informations concernant l'installation, la gestion et l'utilisation de ce produit.

Initiation

Présentation générale d'IBM WebSphere DataPower XC10 Appliance
Démarrage rapide : Installation du matériel du dispositif
2.5+ Tutoriel : Démarrer avec des applications de grille de données simples
Nouveautés de la version 2.5
Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance
Mise à jour de WebSphere DataPower XC10 Appliance

Tâches courantes

Configuration de votre dispositif
Configuration des collectivités et des zones
Configuration des grilles de données
Configuration des clients
Administration des grilles de données
Développement d'applications pour accéder à des grilles de données simples
Surveillance
Sécurité

Traitement des incidents et support

Traitement des incidents
Portail d'assistance
Fix central
Forum
Notes sur l'édition
Page d'accueil du service de support logiciel IBM

Informations complémentaires

Formation à IBM WebSphere DataPower XC10 Appliance
Articles
Redbooks
Communauté dédiée au cache mémoire redimensionnable (cache élastique) d'IBM

Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Le produit sous licence IBM® WebSphere DataPower XC10 Appliance est un dispositif spécialement conçu et optimisé pour offrir une mise en cache simple, rapide et rentable pour les applications WebSphere.

Avantages

DataPower XC10 Appliance offre les avantages suivants :

Evolutivité en toute simplicité

DataPower XC10 Appliance contient une grille de données flexible de 240 Go que vous pouvez utiliser pour héberger les données de vos applications vitales. Pour ajouter davantage de mémoire à la grille de données, vous pouvez ajouter un autre dispositif à la configuration, créant ainsi une collectivité de dispositifs pour l'hébergement de vos données.

Utilisation simple et instantanée sans modification de code

Vous pouvez utiliser DataPower XC10 Appliance sans apporter de modification au code de vos applications. Vous pouvez intégrer directement le dispositif dans les scénarios suivants :

- Gestion de session pour les requêtes HTTP
- Prise en charge de la mémoire cache dynamique WebSphere Application Server

Tolérance aux pannes

Les données des grilles de données sont automatiquement répliquées afin de diminuer le risque de perte.

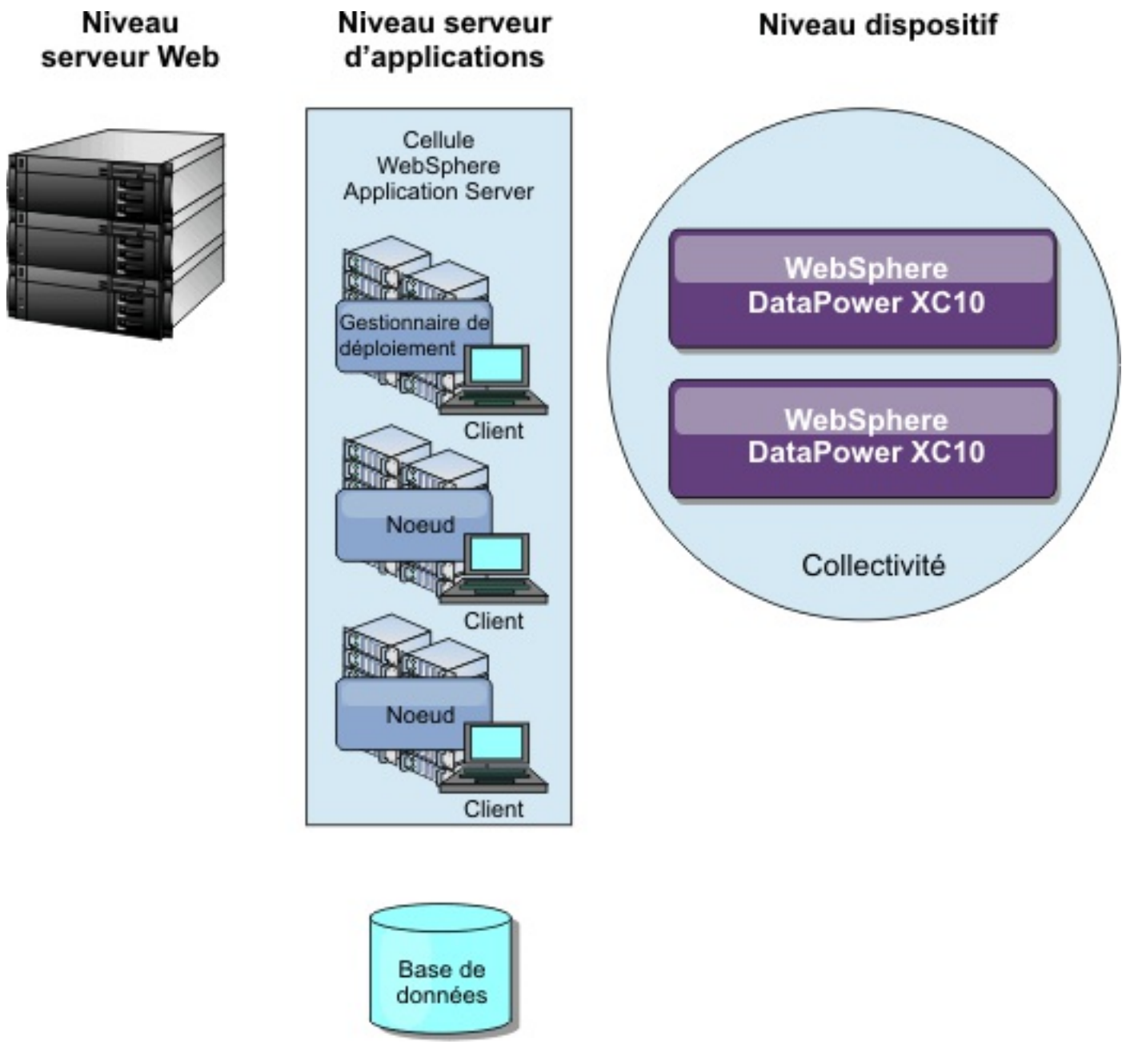
Gestion des utilisateurs simple et flexible

L'interface utilisateur permet de gérer facilement les utilisateurs et les groupes d'utilisateurs de votre dispositif. Elle permet également de créer, de gérer et de contrôler les grilles de données.

Topologie globale

DataPower XC10 Appliance est configuré derrière le niveau serveur d'applications. Au niveau serveur d'applications, vous installez le WebSphere eXtreme Scale Client sur chaque noeud, y compris au niveau du gestionnaire de déploiement. Ce client active la communication entre le niveau serveur d'applications et le niveau dispositif. Au niveau dispositif, vous pouvez regrouper les dispositifs afin de créer une collectivité.

Figure 1. Topologie globale



Nouveautés de la version 2.5

version 2.5 offre des améliorations du contrôle et de la sécurité et prend en charge les grilles de données d'entreprise, les applications .NET, l'interrogation continue et l'invalidation du cache local.

Notes sur l'édition

Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

Topologie du dispositif : collectivités, zones et grilles de données

Une *grille de données* est une unité de stockage qui peut être créée pour stocker les objets d'une application ou d'un ensemble d'applications spécifique. Une *collectivité* les dispositifs afin d'en faciliter l'évolutivité et la gestion. Une *zone* définit un emplacement physique pour votre dispositif et permet de déterminer où les données de la mémoire cache doivent être placées.

2.5+ Présentation de la grille de données d'entreprise

Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Avec le nouveau transport et le nouveau format de sérialisation, vous pouvez connecter des clients Java™ et .NET à une même grille de données.

Présentation du traitement des transactions

WebSphere eXtreme Scale Client utilise des transactions comme mécanisme d'interaction avec les données.

Remarques

Remarques sur les règles de confidentialité

Nouveautés de la version 2.5

version 2.5 offre des améliorations du contrôle et de la sécurité et prend en charge les grilles de données d'entreprise, les applications .NET, l'interrogation continue et l'invalidation du cache local.

WebSphere eXtreme Scale Client for .NET
Installer WebSphere eXtreme Scale Client for .NET vous permet de déployer des applications .NET qui accèdent à la grille de données. Pour en savoir plus...

Etat de démarrage du dispositif
Vous pouvez désormais visualiser l'état de démarrage du dispositif dans l'interface utilisateur. Pour en savoir plus...

Fournisseur de stockage d'état de session ASP.NET
Vous pouvez configurer vos applications ASP.NET pour stocker l'état de session dans la grille de données. Pour en savoir plus...

Interrogation continue
Lorsque vous développez des applications client qui interagissent avec la grille de données, vous pouvez avoir besoin de requêtes qui extraient automatiquement des résultats en temps réel lorsque de nouvelles entrées sont insérées ou mises à jour. Vous pouvez utiliser l'interrogation continue pour être notifié dans votre machine JVM (Java™ virtual machine) lorsque des données sont insérées ou mises à jour dans la grille de données. Cette fonction facilite la gestion des données et de la grille pour les développeurs, les administrateurs, etc. Pour en savoir plus...

Suppression de tâches
Vous pouvez supprimer toutes les tâches dans l'interface utilisateur. Pour en savoir plus...

Mise à jour dynamique des propriétés du client pour .NET
Vous pouvez configurer WebSphere eXtreme Scale Client for .NET pour qu'il détecte de façon dynamique les modifications manuelles apportées aux valeurs de propriété dans les fichiers de propriétés des clients. Il n'est pas nécessaire de redémarrer la connexion à la grille de données pour que ces modifications prennent effet. Pour en savoir plus...

Activation de la norme FIPS
Vous pouvez configurer la collectivité de dispositifs pour qu'elle utilise la norme FIPS (Federal Information Processing Standard) 140-2 pour toutes les communications réseau chiffrées. Cette norme garantit une protection élevée des données transmises. Pour en savoir plus...

Codage de la propriété credentialGeneratorProps dans les fichiers de propriétés de client pour .NET
Vous pouvez coder la valeur de la propriété credentialGeneratorProps dans le fichier `Client.Net.properties` à l'aide de l'utilitaire **`FilePasswordEncoder`**. Pour en savoir plus...

Grille de données d'entreprise
Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Vous pouvez ainsi connecter des clients Java et .NET à une même grille de données. Pour en savoir plus...

eXtreme Data Format (XDF)
XDF repose sur le plug-in MapSerializerPlugin et il est désormais la technologie de sérialisation par défaut utilisée lorsque vous exécutez IBM® eXtremeIO (XIO).

Exemple d'initiation
Vous pouvez utiliser l'exemple d'initiation pour exécuter des applications client Java et .NET qui accèdent aux grilles de données de votre dispositif. Pour en savoir plus...

Alias de grille
Vous pouvez utiliser la passerelle REST pour créer et gérer un alias de grille. Les alias de grille sont utiles lorsque vous devez remplir plusieurs grilles simultanément. Pour en savoir plus...

Gestion des noms d'hôte
La propriété **`publishHost`** vous permet de configurer et de remplacer le nom d'hôte publié. Vous disposez ainsi d'un contrôle plus fin sur la communication entre les clients et les serveurs, ce qui vous permet d'optimiser la communication entre les noeuds dans la grille de données. Pour en savoir plus...

LDAP sur SSL
Sécurisez votre annuaire LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs qui

accèdent à votre dispositif. Pour ce faire, configurez le dispositif pour l'authentification LDAP via une connexion SSL. Pour en savoir plus...

Invalidation du cache local

Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée sur la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Pour en savoir plus...

Nouvelles commandes et nouveaux paramètres de l'utilitaire `xscmd`

- Commande **`xscmd -c getNotificationFilter`** : exécutez cette commande pour afficher les filtres en cours des nouvelles notifications depuis Message Center. Pour en savoir plus...
- Commande **`xscmd -c listenForNotifications`** : exécutez cette commande pour écouter les nouvelles notifications depuis Message Center. Pour en savoir plus...
- Commande **`xscmd -c setNotificationFilter`** : exécutez cette commande pour créer un filtre pour les nouvelles notifications depuis Message Center. Pour en savoir plus...
- Commande **`xscmd -c showLinkedDomains`** : exécutez cette commande pour identifier les domaines de service de catalogue liés au domaine de service de catalogue local. Pour en savoir plus...
- Commande **`xscmd -c showNotificationHistory`** : exécutez cette commande pour afficher la sortie de l'historique des notifications d'événements dans un tableau.
- Commande **`xscmd -c showSessionSize`** : exécutez cette commande pour afficher la taille d'une session. Pour en savoir plus...
- Paramètre **`-to`** ou **`--timeout`** : définissez ce paramètre pour réduire le délai d'attente afin d'éviter d'attendre pendant toute la durée des délais d'attente du système d'exploitation ou du réseau au cours d'un arrêt réseau ou d'une perte du système. Pour en savoir plus...
- Paramètre **`-hc`** ou **`--linkHealthCheck`** : utilisez ce paramètre avec la commande **`xscmd -c showLinkedPrimaries`** pour vérifier si les fragments primaires disposent du nombre approprié de liens de collectivité.
- Commande **`xscmd -c listDisabledForPlacement`** : exécutez cette commande pour afficher la liste des conteneurs de fragments désactivés pour le placement des fragments.
- Commande **`xscmd -c listIndoubts`** : exécutez cette commande pour afficher la liste des transactions en attente de validation. Exécutez cette commande pour résoudre les exceptions potentielles de délai d'attente de verrouillage sur une partition. Pour en savoir plus...
- Commandes **`xscmd -c showReplicationState`** et **`xscmd -c showDomainReplicationState`** : exécutez ces commandes pour afficher l'état des révisions précédentes sur les serveurs de catalogue ou les domaines de service de catalogue. Pour en savoir plus...
- Commande **`xscmd -c showTransport`** : exécutez cette commande pour afficher le type de transport du domaine de service de catalogue. Pour en savoir plus...

Remplacement de la propriété `requestRetryTimeout` dans les grilles de données de cache dynamique

Utilisez la propriété de mémoire cache dynamique **`com.ibm.websphere.xs.dynacache.request_retry_timeout_override`** pour remplacer la propriété **`requestRetryTimeout`** du fichier de propriétés du client et spécifier pendant combien de temps (en millisecondes) une requête peut s'exécuter avant d'expirer. Pour en savoir plus...

Exécution de l'utilitaire `xscmd` à partir de l'interface de ligne de commande du dispositif

Vous pouvez exécuter l'utilitaire **`xscmd`** à partir de l'interface de ligne de commande du dispositif. Dans ce cas, vous n'avez pas besoin de fournir les arguments de ligne de commande pour la sécurité et les connexions de serveur de catalogue. Pour en savoir plus...

Génération de clé secrète

Remplacement de la clé secrète d'authentification définie par défaut en usine par une clé secrète unique Pour en savoir plus...

Mise à jour des statistiques SNMP (Simple Network Monitoring Protocol)

Les bases d'informations de gestion (MIB) fournies avec le WebSphere DataPower XC10 Appliance pour définir les données SNMP disponibles pour le client SNMP ont été mises à jour avec de nouvelles statistiques. Ces dernières incluent applianceUsedBytes, applianceCapacity, gridUsedBytes et gridCapacity. Pour plus d'informations sur l'activation de la surveillance de SNMP et le téléchargement de ce fichier MIB, voir Activation de la surveillance SNMP du dispositif.

Définition du délai d'attente et du délai d'attente pour les requêtes

Vous pouvez spécifier la durée maximum de traitement d'une connexion ou d'une requête, que vous utilisiez le transport eXtremeIO (XIO) ou le transport Object Request Broker (ORB). Par exemple, dans le cas de XIO, vous pouvez utiliser la propriété **`xiotimeout`** pour spécifier le délai d'attente pour les tentatives d'établissement d'une connexion socket sortante, et la propriété **`xioRequestTimeout`** pour spécifier le

nombre de secondes pendant lequel une requête peut attendre avant d'expirer. La propriété **`requestRetryTimeout`** s'applique au transport XIO ou ORB. Elle vous permet de spécifier pendant combien de temps le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Pour en savoir plus...

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Notes sur l'édition

Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

- Dernières mises à jour, limitations et problèmes connus
- Accès à la configuration système et logicielle requise
- Accès à la documentation du produit
- Accès au site de support technique du produit
- Contacter le service de support logiciel IBM

Dernières mises à jour, limitations et problèmes connus

Les notes sur l'édition sont disponibles sur le site de support technique du produit sous forme de notes techniques. Pour afficher une liste de toutes les notes techniques de WebSphere DataPower XC10 Appliance, accédez à la page Web du support technique. Cliquer sur les liens indiqués ici générera une recherche dans la page Web du support des notes sur l'édition correspondantes, lesquelles seront retournées sous forme de liste.

- Pour afficher une liste des notes sur l'édition, accédez à la page Web du support technique.

Accès à la configuration système et logicielle requise

La configuration matérielle et logicielle requise est détaillée dans les pages suivantes :

- Configuration requise

Accès à la documentation du produit

Pour accéder à l'ensemble des informations, accédez à la page Bibliothèque.

Accès au site de support technique du produit

Pour rechercher les informations de support technique et notamment les dernières notes techniques, les fichiers à télécharger et les correctifs, accédez à la page du support technique.

Contacter le service de support logiciel IBM

Si un incident survient lors de l'utilisation du produit, essayez tout d'abord d'effectuer les opérations suivantes :

- Suivez les étapes décrites dans la documentation du produit
- Recherchez la documentation connexe dans l'aide en ligne
- Recherchez les messages d'erreur dans le document de référence des messages

Si vous ne parvenez pas à résoudre l'erreur à l'aide d'une des méthodes précédentes, prenez contact avec le support technique IBM®.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance
Rubrique parent : Traitement des incidents

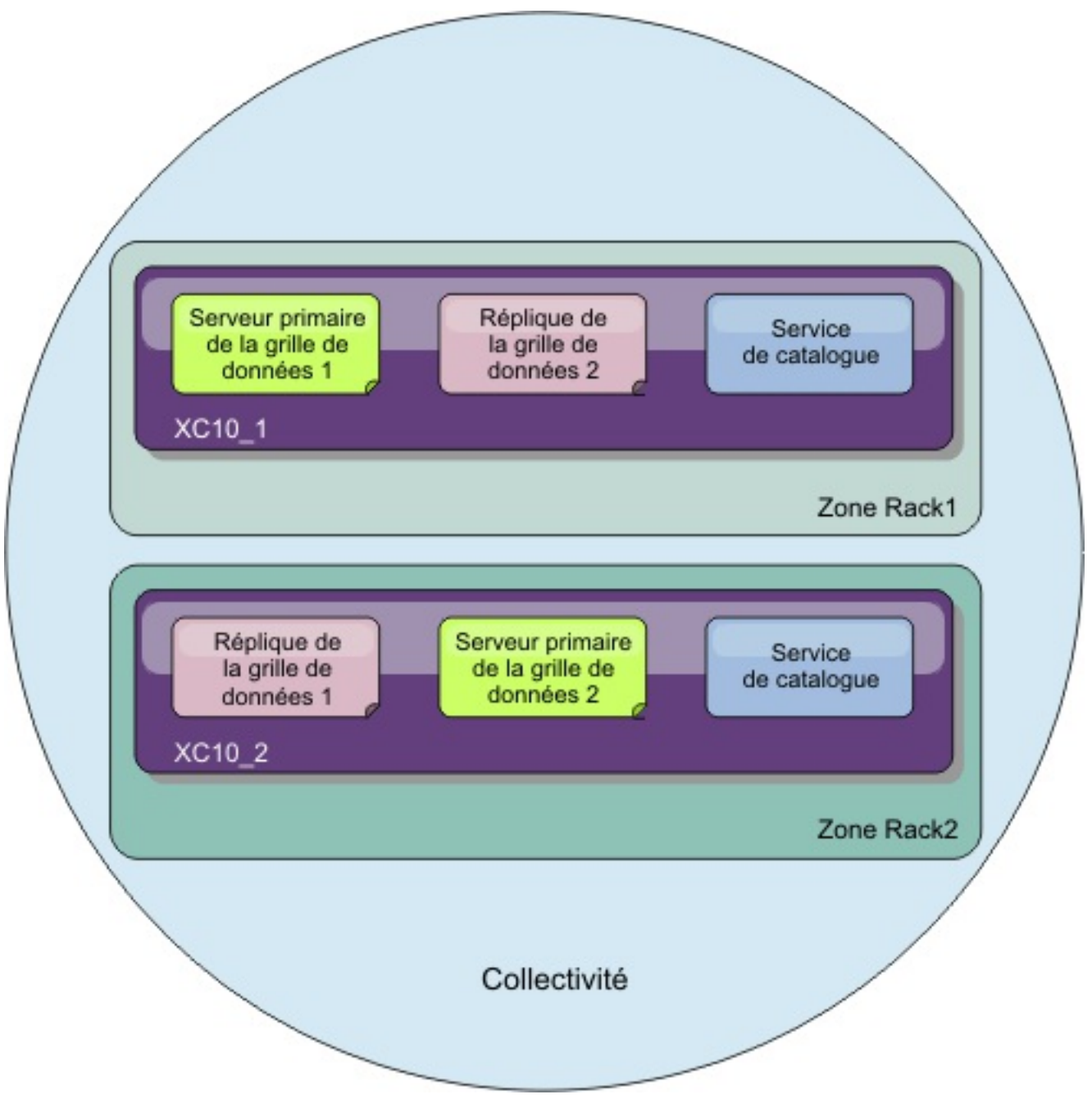
Topologie du dispositif : collectivités, zones et grilles de données

Une *grille de données* est une unité de stockage qui peut être créée pour stocker les objets d'une application ou d'un ensemble d'applications spécifique. Une *collectivité* les dispositifs afin d'en faciliter l'évolutivité et la gestion. Une *zone* définit un emplacement physique pour votre dispositif et permet de déterminer où les données de la mémoire cache doivent être placées.

Topologie du dispositif

Les collectivités et les zones sont associées à une ou plusieurs instances de WebSphere DataPower XC10 Appliance. Chaque dispositif peut être membre d'une collectivité et d'une zone. Chaque dispositif héberge plusieurs grilles de données, qui contiennent les données en mémoire cache.

Figure 1. Topologie des collectivités et des zones



Important : Deux dispositifs sont nécessaires pour rendre la grille de données hautement disponible.

Collectivités et réplication multimaître

La réplication multimaître est une technique qui permet de garantir une disponibilité continue dans plusieurs environnements de déploiement. Des topologies multimaîtres peuvent être mises en oeuvre dans WebSphere DataPower XC10 Appliance en plusieurs collectivités et en les associant. Lorsque vous définissez une collectivité, les informations suivantes sont partagées entre les dispositifs de la collectivité : grilles de données, informations de contrôle, membres de la collectivité et de zone et utilisateurs. Lorsque vous mettez à jour ces informations, les modifications apportées affectent tous les autres dispositifs de la collectivité. Le *service de catalogue* permet la communication entre les dispositifs. Le service de catalogue est un groupe de serveurs de catalogue. Les différents dispositifs de la collectivité exécutent un serveur de catalogue, dans la limite de trois serveurs de catalogue par collectivité. Si vous disposez de plus de trois dispositifs au sein d'une collectivité, le service de catalogue s'exécute sur les trois premiers dispositifs ajoutés à la collectivité. Si vous supprimez un dispositif associé à un serveur de catalogue de la collectivité, ou si un serveur de catalogue devient disponible, le dispositif suivant ajouté à la collectivité exécute le serveur de catalogue. Ce dernier ne bascule pas vers les autres dispositifs.

Pour ajouter un dispositif à une collectivité, ajoutez le nom d'hôte et la clé secrète du dispositif au panneau de configuration de la collectivité à partir d'un autre dispositif. Cette configuration peut s'effectuer à partir de tous les dispositifs de la collectivité. En effet, l'appartenance à la collectivité est conservée pour tous les membres de la collectivité.

Les dispositifs peuvent appartenir à une seule collectivité. Vous ne pouvez pas ajouter un dispositif appartenant déjà à une collectivité à une autre collectivité. Il est également impossible de regrouper deux

collectivités dans une collectivité unique. Pour joindre les dispositifs de différentes collectivités, vous devez supprimer ces dispositifs de leur collectivité respective et les définir ainsi en tant que dispositifs autonomes. Vous pouvez ensuite créer une collectivité regroupant l'ensemble de ces dispositifs.

Vous pouvez utiliser une collectivité pour effectuer la plupart des modifications de configuration. Vous devez toutefois vous connecter à un dispositif donné pour modifier les paramètres des panneaux **Dispositifs** > **Paramètres de dispositif** et **Dispositifs** > **Identification et résolution des incidents**.

Zones

Les zones sont associées à l'emplacement physique du dispositif (par exemple, une ville ou un emplacement d'armoire au sein d'un lab). Les zones permettent au service de catalogue de définir l'emplacement de stockage des données dans vos grilles de données. Par exemple, si les données principales de la grille de données sont stockées dans une zone donnée, les données secondaires sont stockées sur un dispositif situé dans une zone différente. Dans cette configuration, un basculement peut se produire entre un système principal et une réplique si le dispositif contenant les données principales de grille de données tombe en panne.

Grilles de données

Grilles de données regroupe les objets correspondant aux applications. En plaçant des objets en mémoire cache, vous pouvez augmenter les performances de vos applications. Il existe trois types de grilles de données :

grille de données simple

Les grilles de données simples regroupent les données sous forme de paires clé-valeur. Par exemple, vous pouvez stocker les résultats d'une requête de base de données dans une grille de données simple. L'implémentation d'une grille de données simple s'effectue à l'aide de l'API ObjectMap. Le fonctionnement de l'API ObjectMap est similaire à celui des mappes Java™.

grille de données de session

Si vous utilisez des sessions WebSphere Application Server, vous pouvez configurer votre application de manière à utiliser une grille de données de session sur le dispositif des données de gestion de session. Vous pouvez configurer votre application de manière à utiliser une grille de données de session lorsque vous installez une nouvelle application. Vous pouvez également mettre à jour les paramètres de vos applications ou de vos serveurs existants de manière à utiliser une grille de données de session sur le dispositif.

grille de données de mémoire cache dynamique

Une grille de données de mémoire cache dynamique définie sur un dispositif permet de stocker des données en provenance de la mémoire cache dynamique de WebSphere Application Server. Vous pouvez activer des applications rédigées à l'aide de l'API de cache dynamique ou les applications utilisant la mise en cache au niveau du conteneur ,par exemple, les servlets, pour l'utilisation du dispositif en tant que fournisseur de mémoire cache. Les serveurs d'applications utilisent dès lors une quantité de mémoire inférieure. Toutes les données de mémoire cache sont transférées vers le dispositif et disparaisse de la mémoire des serveurs d'applications.

Répliques de grilles de données

Vous pouvez définir un nombre cible de répliques pour une grille de données spécifique. Des répliques sont créées lorsque la collectivité contient au moins deux dispositifs. Si elle ne contient qu'un seul dispositif, aucune réplique n'est créée. Si vous avez un nombre de dispositifs *n* dans votre collectivité, le nombre maximal de répliques est *n-1* car un des dispositifs héberge la grille de données principale. Si votre nombre cible de répliques est supérieur à la valeur *n-1* actuelle, davantage de répliques peuvent être placées lorsque vous ajoutez des dispositifs à la collectivité. Envisagez de définir le nombre de répliques au nombre de répliques le plus élevé que vous pourrez souhaiter dans le futur. La modification des paramètres des répliques requiert l'effacement des grilles de données : définissez donc la valeur en tenant compte du nombre futur de répliques. Quand de nouveaux dispositifs rejoignent la collectivité, des répliques supplémentaires sont créées. Les grilles de données principales et répliques sont réparties de façon homogène, ou segmentées, sur tous les dispositifs de la collectivité. Quand de nouveaux dispositifs rejoignent la collectivité, un nouvel équilibrage a lieu pour répartir les grilles de données principales et répliques.

Les répliques peuvent être synchrones ou asynchrones. Les répliques synchrones reçoivent des mises à jour dans le cadre de la transaction sur la grille de données principale. Les répliques asynchrones sont mises à jour après la validation de la transaction sur la grille de données principale. Les répliques synchrones garantissent la cohérence des données. En contrepartie, elles augmentent le délai de traitement des requêtes par rapport aux répliques asynchrones. Les répliques asynchrones n'offrent pas les mêmes garanties en termes de cohérence des données, mais accélèrent les délais de traitement des transactions. Une grille de données possède une réplique asynchrone par défaut. Un algorithme de placement contrôle l'emplacement des répliques.

Mappes

Les mappes sont les structures de données qui contiennent les données de la grille de données en paires clé-valeur. Une grille de données unique peut avoir plusieurs mappes résidant dans les grilles de données et les répliques de grille de données.

Vous pouvez créer des mappes supplémentaires dans la grille de données en demandant à votre application client de se connecter à une mappe spécifiquement nommée. Une mappe dynamique est automatiquement créée.

Liaisons de collectivité

Une seule collectivité ne doit pas couvrir un réseau non fiable car des incidents de faux positif risquent d'être détectés. Cependant, vous pouvez être amené à répliquer des données de grille de données sur les dispositifs dont la connectivité du réseau n'est pas fiable. Voici quelques scénarios courants dans lesquels vous pouvez être amené à utiliser ce type de topologie :

- Reprise après incident entre des centres de données dans lesquels une collectivité est active et une autre est utilisée à des fins de secours
- Centres de données géographiquement répartis dans lesquels toutes les collectivités sont actives pour les clients géographiquement proches

Une fois que vous connectez deux collectivités, toutes les grilles de données portant le même nom sont répliquées de façon asynchrone d'une collectivité à l'autre. Ces grilles de données doivent comporter le même nombre de répliques dans chaque collectivité et posséder les mêmes configurations de mappe dynamique.

> **Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître**
> Vous disposez de plusieurs options pour choisir la topologie de votre déploiement qui intègre plusieurs collectivités. Des topologies multimaîtres peuvent être mises en oeuvre dans DataPower XC10 Appliance en plusieurs collectivités et en les associant.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

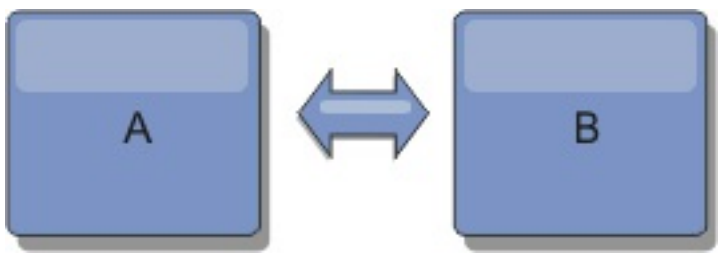
Tâches associées:
Configuration d'une réplication multimaître entre les collectivités

Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître

Vous disposez de plusieurs options pour choisir la topologie de votre déploiement qui intègre plusieurs collectivités. Des topologies multimaîtres peuvent être mises en oeuvre dans DataPower XC10 Appliance en plusieurs collectivités et en les associant.

Liaisons connectant les collectivités

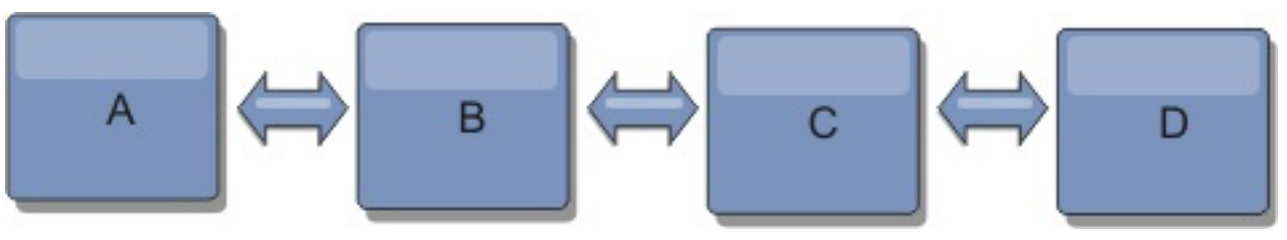
Une infrastructure de grilles de données de réplication est un graphique de collectivités interconnectés avec des liaisons bidirectionnelles. Avec une liaison, deux collectivités peuvent communiquer les modifications de données. Par exemple, la topologie la plus simple est une paire de collectivités avec une liaison unique entre eux. Les collectivités sont nommés par ordre alphabétique: A, B, C, etc., à partir de la gauche. Une liaison peut traverser un réseau WAN (wide area network) pour couvrir une grande distance. Même si la liaison est interrompue, vous pouvez toujours modifier les données dans une collectivité. La topologie rapproche les modifications quand la liaison reconnecte les collectivités. Les liaisons tentent automatiquement de se reconnecter si la connexion réseau est interrompue.



Après avoir établi les liaisons, le produit tente d'abord de rendre chaque collectivité identique. Ensuite, eXtreme Scale tente de maintenir identiques les conditions à mesure que des modifications se produisent dans un collectivité. L'objectif vise à faire de chaque collectivité le miroir exact d'un autre collectivité connecté par les liaisons. Les liaisons de réplication entre les collectivités permettent de copier une modification effectuée dans un collectivité vers les autres collectivités.

Topologies linéaires

Même s'il s'agit d'un déploiement simple, une topologie linéaire montre certaines qualités des liaisons. Tout d'abord, il n'est pas nécessaire qu'un collectivité soit directement connecté à chacun des autres domaines de service de catalogue pour recevoir des modifications. Le collectivité B extrait les modifications du collectivité A. Le collectivité C reçoit les modifications du collectivité A via le collectivité B, lequel connecte les domaines A et C. De même, le collectivité D reçoit les modifications des autres collectivités via le collectivité C. Cette fonction répartit la charge de distribution des modifications en l'éloignant de la source des modifications.



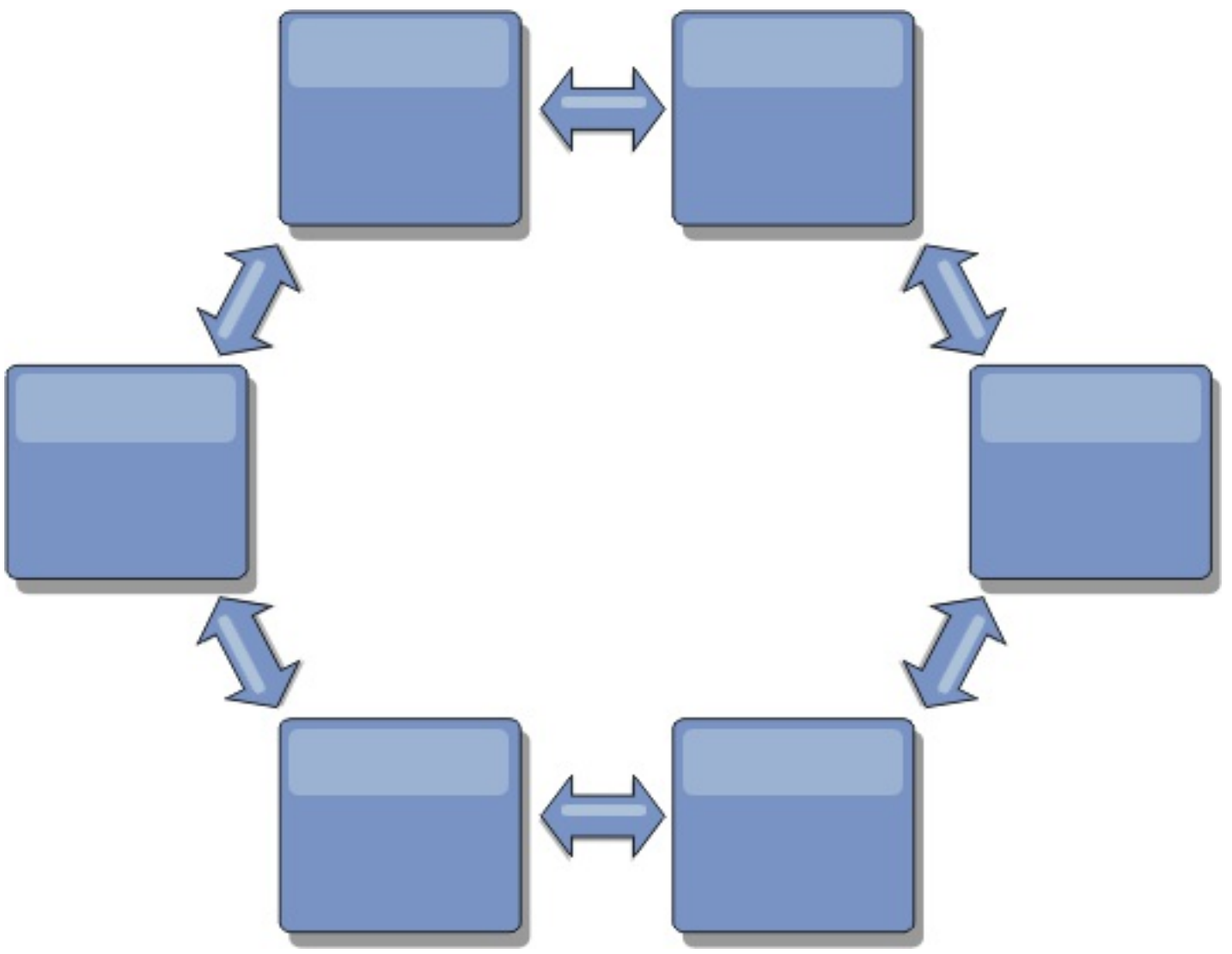
Notez que si le collectivité C est défaillant, les actions suivantes se produisent :

1. Le collectivité D serait orphelin jusqu'au redémarrage du collectivité C.
2. Le collectivité C doit se synchroniser avec le collectivité B, lequel est une copie du collectivité A.
3. Le collectivité D utilise le collectivité C pour se synchroniser avec les modifications des domaines de service de catalogues A et B. Ces modifications se sont produites initialement lorsque le collectivité D étaient orphelin (lorsque le collectivité C était arrêté).

Enfin, les collectivités A, B, C et D, sont de nouveau tous identiques.

Topologies en anneau

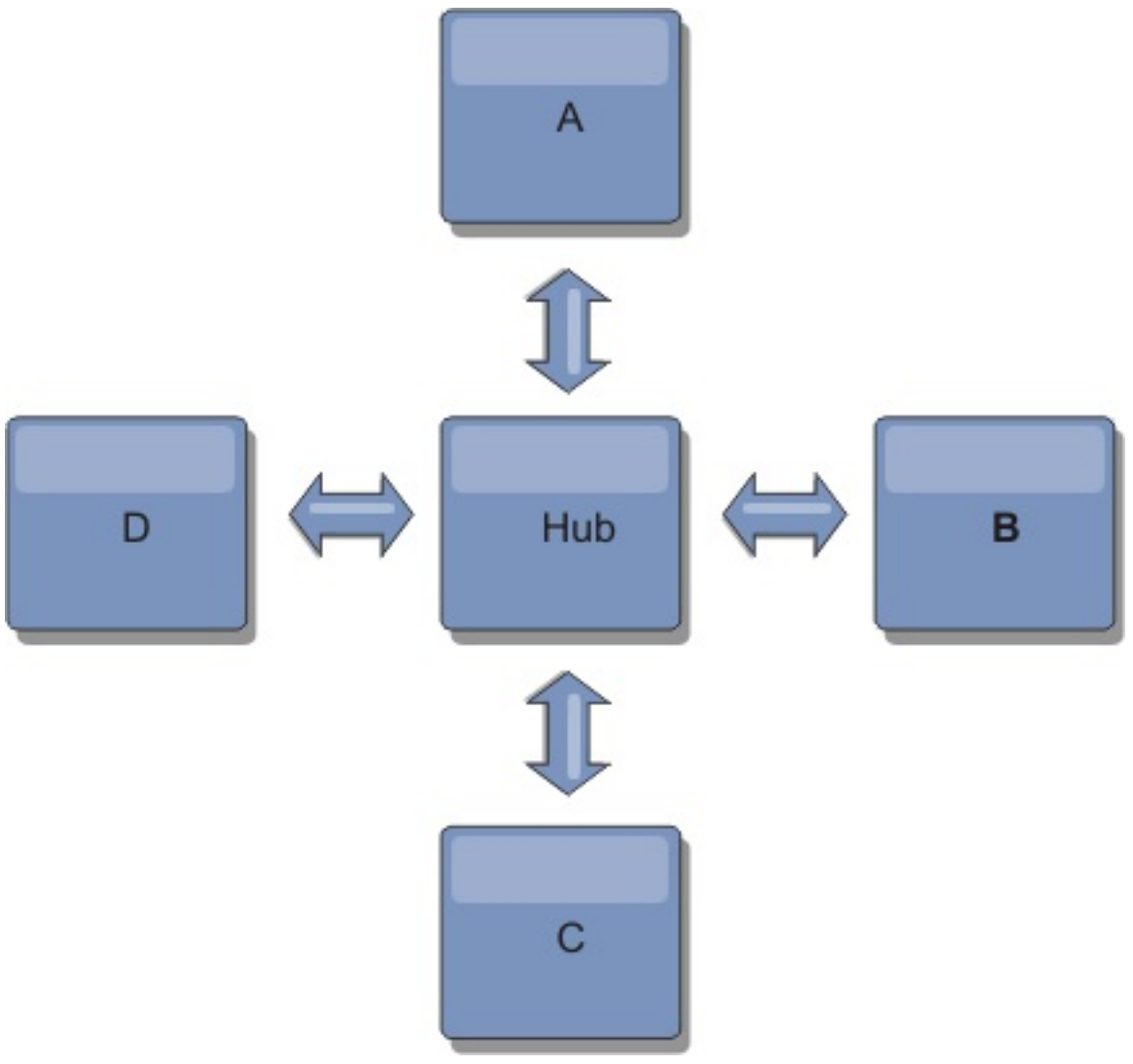
Les topologies en anneau sont un exemple de topologie encore plus résilientes. Lorsqu'un collectivité ou une liaison unique est défaillant, les collectivités restants peuvent encore obtenir des modifications. Les collectivités parcourent l'anneau en s'éloignant de la défaillance. Chaque collectivité possède au maximum deux liens vers d'autres collectivités, quelle que soit la taille de la topologie en anneau. Le délai de propagation des modifications peut être important. Les modifications d'un collectivité particulier peuvent devoir traverser plusieurs liaisons pour que tous les collectivités aient les modifications. Une topologie linéaire a la même caractéristique.



Vous pouvez également déployer une topologie en anneau plus sophistiquée, avec un collectivité racine au centre de l'anneau. Le collectivité racine fait office de point central de rapprochement. Les autres collectivités font office de points distants de rapprochement pour les modifications se produisant dans le collectivité racine. Le collectivité racine peut arbitrer les modifications entre les collectivités. Si une topologie en anneau contient plusieurs anneaux autour d'un collectivité racine, le collectivité ne peut pas arbitrer les modifications dans la partie interne de l'anneau. Toutefois, les résultats de l'arbitrage sont propagés dans les collectivités des autres anneaux.

Topologies en étoile

Avec une topologie en étoile, les modifications parcourent un collectivité du concentrateur. Etant donné que le concentrateur est le seul collectivité intermédiaire spécifié, les topologies en étoile ont une latence inférieure. Le collectivité du concentrateur est connecté à chaque branche de collectivité via une liaison. Le concentrateur répartit les modifications entre les collectivités. Il fait office de point de rapprochement pour les collisions. Dans un environnement soumis à une fréquence élevée de modifications, le concentrateur peut avoir besoin de s'exécuter sur plus de matériels que les branches pour rester synchronisé. WebSphere DataPower XC10 Appliance est conçu pour évoluer de manière linéaire, ce qui signifie que l'on peut, si nécessaire, étoffer le concentrateur sans difficultés. Toutefois, si le concentrateur tombe en panne, les modifications ne sont pas distribuées jusqu'à ce qu'il redémarre. Toutes les modifications sur les branches du sous-domaine de service de catalogue seront réparties après la reconnexion du concentrateur.



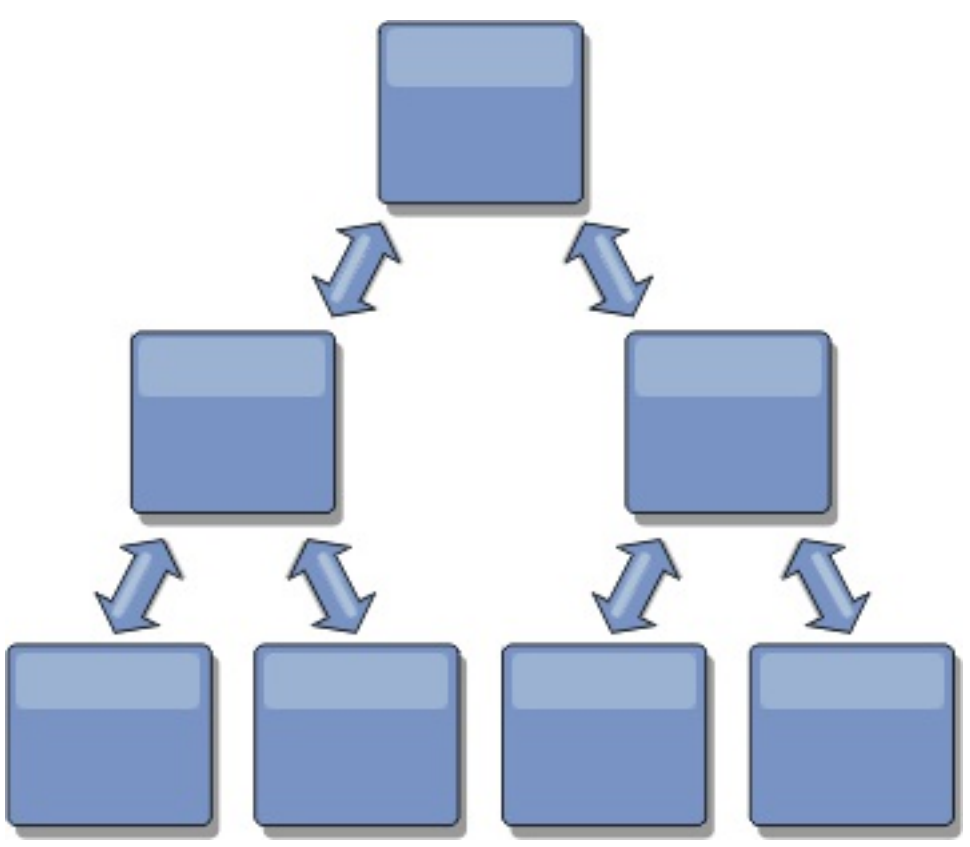
Vous pouvez également utiliser une stratégie avec les clients intégralement répliqués, une variante de la topologie qui utilise une paire de serveurs s'exécutant comme concentrateur. Chaque client crée une grille de données à conteneur unique autonome avec un catalogue dans la machine virtuelle Java client. Un client utilise sa grille de données pour se connecter au catalogue du concentrateur. Cette connexion provoque la synchronisation du client avec le concentrateur dès que le client obtient une connexion au concentrateur.

Toutes les modifications effectuées par le client sont locales pour le client et elles sont répliquées vers le concentrateur de manière asynchrone. Le concentrateur joue le rôle de collectivité d'arbitrage, répartissant les modifications à tous les clients connectés. La topologie de clients intégralement répliqués fournit un cache L2 fiable pour un associateur relationnel d'objets comme OpenJPA. Les modifications sont réparties rapidement via le concentrateur entre les machines virtuelles client. Si la taille du cache peut être contenue dans le segment de mémoire disponible, la topologie est une architecture fiable pour ce style de cache L2.

Si nécessaire, utilisez plusieurs partitions pour échelonner le collectivité concentrateur sur plusieurs machines virtuelles Java. Etant donné que toutes les données doivent toujours tenir sur une seule machine virtuelle Java client, plusieurs partitions augmentent la capacité du concentrateur à répartir et à arbitrer les modifications. Cependant, plusieurs partitions ne changent pas la capacité d'un collectivité unique.

Topologies en arbre

Vous pouvez également utiliser un arbre dirigé acyclique. Un arbre acyclique n'a pas de cycles ou de boucles, et une configuration dirigée limite les liaisons aux parents et enfants existants uniquement. Cette configuration est utile pour les topologies disposant d'un grand nombre de collectivités. Dans ces topologies, il n'est pas pratique d'avoir un concentrateur central connecté à chaque branche. Ce type de topologie peut également être utile lorsque vous devez ajouter des collectivités enfant sans mettre à jour le collectivité racine.



Une topologie en arbre peut toujours avoir un point central de rapprochement dans le collectivité racine. Le deuxième niveau peut toujours fonctionner en tant que point de rapprochement distant pour les modifications se produisant dans le collectivité en dessous. Le collectivité racine peut arbitrer les modifications entre les collectivités sur le deuxième niveau uniquement. Vous pouvez également utiliser des arbres n-aire ayant chacun n enfants à chaque niveau. Chaque collectivité se connecte à *n* liaisons.

Clients intégralement répliqués

Cette variante de la topologie implique une paire de serveurs s'exécutant comme concentrateur. Chaque client crée une grille de données à conteneur unique autonome avec un catalogue dans la machine virtuelle Java client. Un client utilise sa grille de données pour se connecter au catalogue du concentrateur, ce qui provoque la synchronisation du client avec le concentrateur dès que le client obtient une connexion au concentrateur.

Toutes les modifications effectuées par le client sont locales pour le client et elles sont répliquées vers le concentrateur de manière asynchrone. Le concentrateur joue le rôle de collectivité d'arbitrage, répartissant les modifications à tous les clients connectés. La topologie de clients intégralement répliqués fournit un bon cache de niveau 2 pour un associateur relationnel d'objets comme OpenJPA. Les modifications sont réparties rapidement via le concentrateur entre les machines virtuelles client. Tant que la taille du cache peut être contenue dans l'espace de segment mémoire disponible des clients, cette topologie est une architecture tout à fait indiquée pour ce style de cache de niveau 2.

Si nécessaire, utilisez plusieurs partitions pour échelonner le collectivité concentrateur sur plusieurs machines virtuelles Java. Toutes les données devant tenir sur une seule machine virtuelle Java, l'utilisation de partitions multiples augmente la capacité du concentrateur à répartir et à arbitrer les modifications, mais elle ne change pas la capacité d'un collectivité unique.

Rubrique parent : Topologie du dispositif : collectivités, zones et grilles de données

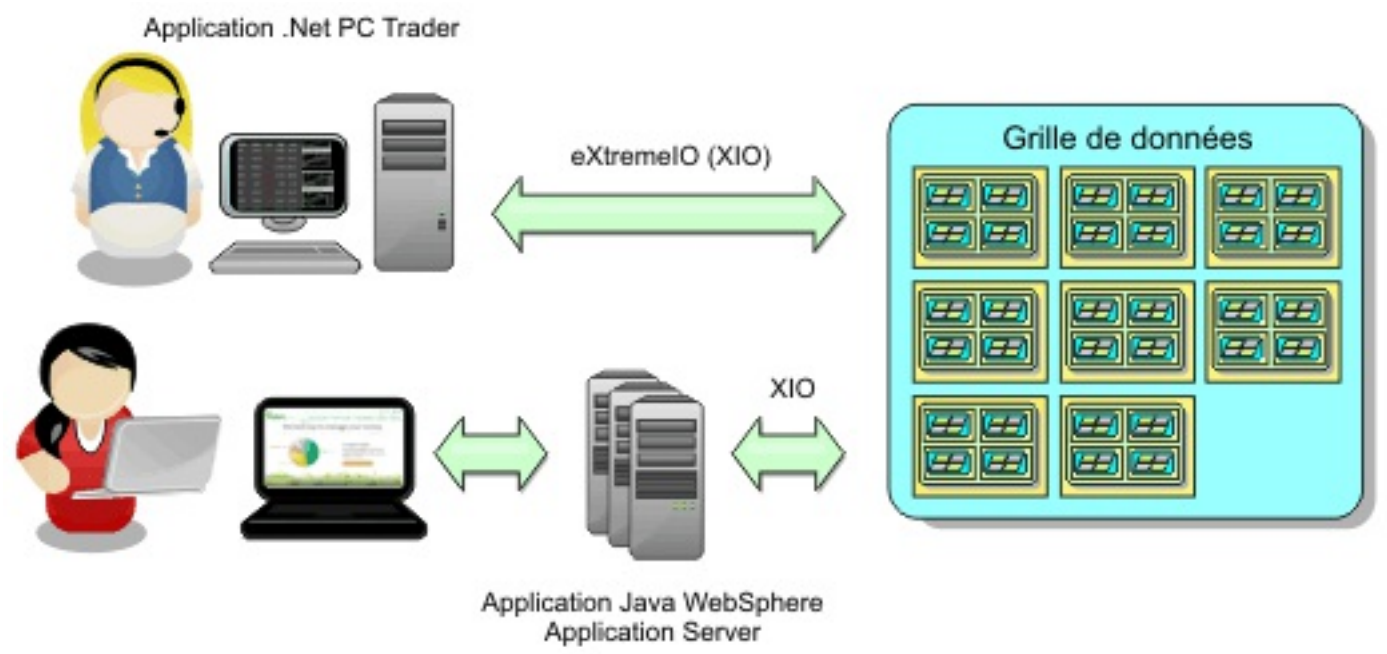
Tâches associées:
Configuration d'une réplication multimaître entre les collectivités

Présentation de la grille de données d'entreprise

Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Avec le nouveau transport et le nouveau format de sérialisation, vous pouvez connecter des clients Java™ et .NET à une même grille de données.

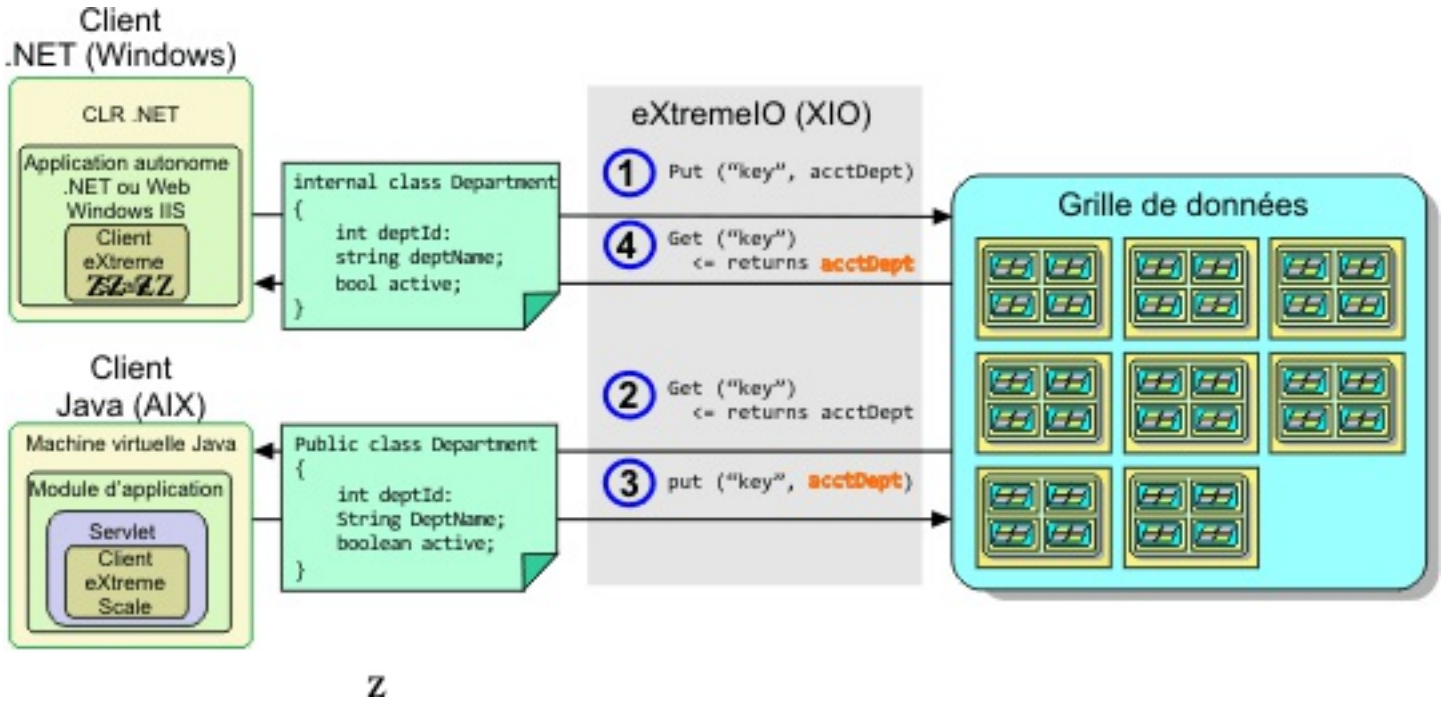
Avec la grille de données d'entreprise, vous pouvez créer plusieurs types d'applications, écrites dans divers langages de programmation, pour accéder aux mêmes objets dans la grille de données. Dans les versions antérieures, les applications de grille de données devaient être écrites en Java uniquement. Avec la fonction de grille d'entreprise, vous pouvez écrire des applications .NET qui créent, extraient, mettent à jour et suppriment des objets de la même grille de données que l'application Java.

Figure 1. Présentation générale de la grille de données d'entreprise



Mises à jour d'objets dans différentes applications

Figure 2. Flux de mise à jour d'un objet dans la grille de données d'entreprise



Z

1. Le client .NET enregistre les données dans son format dans la grille de données.
2. Les données sont stockées dans un format universel pour que lorsque le client Java les demande, elles puissent être converties dans le format Java.
3. Le client Java met à jour et enregistre de nouveau les données.
4. Le client .NET accède aux données mises à jour, période au cours de laquelle les données sont converties dans le format .NET.

Mécanisme de transport

eXtremeIO (XIO) est un protocole de transport multiprotocole. XIO remplace le courtier ORB (Object Request Broker) Java. Avec le courtier ORB, WebSphere DataPower XC10 Appliance est lié aux applications client natives Java. XIO est un mécanisme de transport personnalisé dédié à la mise en cache et qui permet aux applications client dans différents langage de programmation de se connecter à la grille de données.

Format de sérialisation

Le format de données XDF (eXtreme data format) est un format de sérialisation multiplateforme XDF remplace la sérialisation Java dans les mappes dont l'attribut CopyMode a la valeur `C0PY_T0_BYTES` dans le fichier XML descripteur ObjectGrid. XSF améliore les performances et rend les données plus compactes. En outre, XDF permet aux applications client dans différents langages de programmation de se connecter à la

même grille de données.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance
Rubrique parent : Scénario : Configuration d'une grille de données d'entreprise
Rubrique suivante : Configuration d'IBM eXtremeIO (XIO)

Présentation du traitement des transactions

WebSphere eXtreme Scale Client utilise des transactions comme mécanisme d'interaction avec les données.

Java

Traitement des transactions dans les applications Java™

Pour pouvoir interagir avec les données, l'unité d'exécution de votre application a besoin de sa propre session. Lorsque l'application souhaite utiliser ObjectGrid sur une unité d'exécution, appelez l'une des méthodes ObjectGrid.getSession pour obtenir une session. Avec cette session, l'application peut travailler avec les données stockées dans les mappes ObjectGrid.

Lorsqu'une application utilise un objet Session, la session doit être dans le contexte d'une transaction. Une transaction commence et se valide ou commence et s'annule en utilisant les méthodes begin, commit et rollback sur l'objet Session. Les applications peuvent également utiliser le mode de validation automatique, dans lequel la Session initie et valide automatiquement une transaction lorsqu'une opération est effectuée sur la mappe. Ce mode d'auto-validation ne permet pas de regrouper des opérations multiples en une seule transaction. Il s'agit donc de l'option la plus lente si vous créez un lot d'opérations multiples dans une seule transaction. Cependant, pour les transactions contenant une seule opération, l'auto-validation est l'option la plus rapide.

Lorsque l'application n'utilise plus la session, utilisez la méthode facultative Session.close() pour fermer la session. La fermeture de la session a pour effet de libérer cette dernière du segment de mémoire et de permettre de réutiliser des appels ultérieurs vers la méthode getSession(), ce qui améliore les performances.

.NET

Traitement des transactions dans les applications .NET

Pour pouvoir interagir avec les données, chaque unité d'exécution de votre application a besoin de son propre objet de transaction. Pour utiliser l'interface IGrid sur une unité d'exécution de votre application, appelez l'une des méthodes suivantes :

- IGrid.GetGridMapPessimisticAutoTx
- IGrid.GetGridMapPessimisticTx

Lorsque vous appelez ces méthodes, vous obtenez un objet IGridMap qui possède un objet de transaction unique. Avec cet objet IGridMap, l'application peut interagir avec les données stockées dans les mappes IGrid. Lorsqu'une application utilise un objet IGridMapPessimisticTx, les opérations effectuées sur la grille de données doivent se situer dans le contexte d'une transaction. Une transaction commence et se valide ou commence et s'annule en utilisant les méthodes begin, commit et rollback sur l'objet IGridTransaction. Les applications peuvent également utiliser le mode de validation automatique, dans lequel l'IGridMapPessimisticAutoTx initie et valide automatiquement une transaction lorsqu'une opération est effectuée sur la mappe. Ce mode d'auto-validation ne permet pas de regrouper des opérations multiples en une seule transaction. Il s'agit donc de l'option la plus lente si vous créez un lot d'opérations multiples dans une seule transaction. Cependant, pour les transactions contenant une seule opération, l'auto-validation est l'option la plus rapide.

Lorsque votre application n'utilise plus l'instance IGridMap, vous pouvez supprimer l'objet IGridMap. Cette suppression entraîne la fermeture de l'objet de transaction associé. Par conséquent, les appels ultérieurs des méthodes GetGridMapPessimisticAutoTx et GetGridMapPessimisticTx peuvent réutiliser un objet de transaction libre existant, ce qui améliore les performances.

Transactions

Les transactions offrent de nombreux avantages pour le stockage et la manipulation de données. Vous pouvez utiliser des transactions pour protéger la grille de données contre les modifications simultanées, appliquer plusieurs modifications comme unité simultanée, répliquer des données et implémenter un cycle de vie pour les verrous appliqués aux modifications.

Stratégies de verrouillage

Les stratégies de verrouillage disponibles sont les suivantes : pessimiste, optimiste et aucune.

Types de verrou

Lorsque vous utilisez les verrouillages pessimiste et optimiste, des verrous partagés (S), pouvant être mis à niveau (U) et exclusifs (X) sont utilisés pour maintenir la cohérence. Les verrous optimistes n'étant pas maintenus, c'est lorsque vous configurez le verrouillage pessimiste que les effets du verrouillage sont le plus apparents. Les verrous ont des cycles de vie. Les différents types de verrou ne sont pas tous compatibles entre eux de la même manière. Les verrous doivent être traités dans l'ordre approprié pour éviter les situations d'interblocage.

Interblocages
Un interblocage peut se produire lorsque deux transactions tentent de mettre à jour la même entrée de cache.

Accès aux données et transactions
WebSphere eXtreme Scale Client utilise des transactions. Dès lors qu'une application dispose d'une connexion à une grille de données, vous pouvez accéder aux données et interagir avec celles-ci dans la grille.

Isolement des transactions
Vous pouvez utiliser l'un des trois niveaux d'isolement de transaction suivants afin d'optimiser la sémantique de verrouillage qui maintient la cohérence dans chaque mappe de cache : lecture reproductible, lecture validée et lecture non validée.

Java **Validation en deux phases et reprise sur incident**
Le protocole de validation en deux phases coordonne toutes les partitions qui participent à une transaction répartie, en déterminant si la transaction doit être validée ou annulée.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Transactions

Les transactions offrent de nombreux avantages pour le stockage et la manipulation de données. Vous pouvez utiliser des transactions pour protéger la grille de données contre les modifications simultanées, appliquer plusieurs modifications comme unité simultanée, répliquer des données et implémenter un cycle de vie pour les verrous appliqués aux modifications.

Quand une transaction démarre, WebSphere eXtreme Scale Client alloue une mappe spéciale des différences pour contenir les changements en cours ou des copies des paires clé-valeur que la transaction utilise. En règle générale, quand un accès à une paire clé-valeur se produit, la valeur est copiée avant que l'application ne la reçoive. Dans les applications Java™, la mappe des différences assure le suivi de toutes les modifications pour les opérations telles que insert, update, get et remove. Dans les applications .NET, la mappe des différences assure le suivi des modifications pour les opérations add, replace, get et remove. Les clés ne sont pas copiées, car elles sont considérées comme non modifiables. Si une transaction est annulée, les informations de la mappe des différences sont supprimées et les verrous sur les entrées sont libérés. Quand une transaction est validée, les changements sont appliqués à la mappe et les verrous sont libérés.

Java Si un objet ObjectTransformer est spécifié dans une application Java, il est utilisé pour copier la valeur. Si la transaction utilise le verrouillage optimiste, les images précédentes des valeurs sont également suivies afin d'être comparées quand la transaction est validée.

Java Si le verrouillage optimiste est utilisé dans une application Java, WebSphere eXtreme Scale Client compare les versions des images précédentes des valeurs avec les valeurs qui se trouvent dans la mappe. Ces valeurs doivent correspondre pour que la transaction soit validée. Cette comparaison autorise un plan de verrouillage à version multiple, mais au prix de la création de deux copies quand la transaction accède à l'entrée. Toute les valeurs sont à nouveau copiées et la nouvelle copie est stockée dans la mappe. WebSphere eXtreme Scale Client crée cette copie pour se protéger contre le fait que l'application change sa référence à la valeur après validation.

Il est possible de ne pas utiliser plusieurs copies des informations. L'application peut enregistrer une copie en utilisant un verrouillage pessimiste au lieu d'un verrouillage optimiste, au prix d'une limitation des accès simultanés. La copie de la valeur au moment de la validation peut également être évitée si l'application accepte de ne pas changer la valeur après une validation.

.NET **Remarque :** Les applications .NET ne prennent en charge que le verrouillage pessimiste.

Avantages des transactions

Utilisez les transactions pour les raisons suivantes :

En utilisant des transactions, vous pouvez :

- annuler les modifications si une exception se produit ou si la logique application requiert l'annulation des changements d'état,
- appliquer plusieurs changements en tant qu'unité atomique au moment de la validation,
- verrouiller et déverrouiller les données afin d'appliquer des changements multiples en tant qu'unité atomique au moment de la validation,
- protéger une unité d'exécution contre les modifications simultanées,
- implémenter un cycle de vie pour les verrous sur les changements,
- produire une unité de réplication atomique.

Taille des transactions

Les transactions volumineuses sont plus efficaces, en particulier pour la réplication. Cependant, les grandes transactions peuvent avoir un effet néfaste sur les accès simultanés car les verrous sur les entrées sont maintenus plus longtemps. Si vous utilisez de grandes transactions, vous pouvez accroître les performances de réplication. Cette augmentation des performances est importante lors du préchargement d'une mappe. Essayez différentes tailles de lot pour déterminer ce qui fonctionne le mieux pour votre scénario.

Mode de validation automatique

Java Si aucune transaction n'a démarré activement, quand une application interagit avec un objet ObjectMap, une opération automatique de démarrage et de validation est effectuée pour l'application. Cette opération fonctionne mais elle empêche l'annulation et le verrouillage de fonctionner efficacement. La vitesse de réplication synchrone est affectée à cause de la très petite taille des transactions. Si vous utilisez une application de gestion des entités, n'utilisez pas le mode de validation automatique car les objets recherchés avec la méthode EntityManager.find deviennent immédiatement non gérés au retour de la méthode et deviennent inutilisables.

.NET Dans les applications .NET, l'interface de mappe GridMapPessimisticAutoTx fournit les opérations begin et commit automatiques équivalentes. Les limitations sont les mêmes : l'annulation et le verrouillage ne fonctionnent pas correctement et la vitesse de réplication synchrone est réduite.

Java

Intégration des transactions Java EE

WebSphere eXtreme Scale Client comporte un adaptateur de ressources conforme à Java Connector Architecture (JCA) 1.5 qui prend en charge les connexions client à une grille de données distante et la gestion des transactions locales. Les applications Java Platform, Enterprise Edition (Java EE) telles que des servlets, des fichiers JavaServer Pages (JSP) et des composants Enterprise JavaBeans (EJB), peuvent démarquer les transactions WebSphere eXtreme Scale Client à l'aide de l'interface javax.resource.cci.LocalTransaction standard ou de l'interface de session WebSphere eXtreme Scale Client.

Lorsque de l'exécution dans WebSphere Application Server avec le support LPS (Last Participant Support) activé dans l'application, vous pouvez inscrire la transaction WebSphere eXtreme Scale Client dans une transaction globale avec d'autres ressources transactionnelles de validation en deux phases.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Programmation de transactions dans des applications .NET

Stratégies de verrouillage

Les stratégies de verrouillage disponibles sont les suivantes : pessimiste, optimiste et aucune.

Les verrous sont liés aux transactions. Vous pouvez spécifier les paramètres de verrouillage suivants :

Java **Aucun verrouillage**

L'exécution sans verrouillage est la plus rapide. Si vous utilisez des données en lecture seule, vous n'avez peut-être pas besoin de verrouillage.

Restriction : Les mappes de sauvegarde configurées pour utiliser une stratégie de non-verrouillage ne peuvent pas participer à une transaction multipartition.

Java .NET **Verrouillage pessimiste**

Place des verrous sur les entrées, puis les maintient jusqu'au moment de la validation. Cette stratégie offre une bonne cohérence au prix d'une réduction de la rapidité de traitement.

Java **Verrouillage optimiste**

Prend une image avant de chaque enregistrement sur lequel la transaction porte et compare cette image avec les valeurs d'entrées en cours quand la transaction est validée. Si les valeurs d'entrées changent, la transaction est annulée. Aucun verrou n'est maintenu jusqu'au moment de la validation. Cette stratégie de verrouillage offre un meilleur accès simultané que les stratégies pessimistes, au risque que la transaction soit annulée et au prix de la mémoire nécessaire pour une copie supplémentaire de l'entrée.

Important : Si vous utilisez une application client avec WebSphere eXtreme Scale Client for .NET, seul le verrouillage pessimiste est pris en charge.

Gestionnaire de verrous

Lors de l'utilisation d'une stratégie de verrouillage PESSIMISTIC ou OPTIMISTIC, un gestionnaire de verrous est créé pour la mappe de sauvegarde. Il utilise une mappe de hachage pour rechercher les entrées verrouillées par une ou plusieurs transactions. S'il existe plusieurs entrées de mappe dans la mappe de hachage, plus nombreux sont les compartiments de verrouillage, meilleures sont les performances. Le risque de conflit de synchronisation Java™ diminue lorsque le nombre de compartiments augmente. Plus les compartiments de verrouillage sont nombreux, plus les accès simultanés le sont également. Les exemples précédents montrent comment une application peut définir le nombre de compartiments de verrouillage à utiliser pour une instance BackingMap donnée.

Java .NET

Verrouillage pessimiste

La stratégie de verrouillage PESSIMISTIC obtient des verrous pour les entrées de cache et doit être utilisée lorsque les données sont modifiées fréquemment. A chaque lecture d'une entrée de cache, un verrou est obtenu et maintenu de façon conditionnelle jusqu'à la fin de la transaction. La durée de certains verrous peut être paramétrée à l'aide des niveaux d'isolement de transaction pour la session.

La stratégie de verrouillage pessimiste est à utiliser pour les opérations de mappe en lecture et en écriture lorsqu'aucune autre stratégie de verrouillage n'est possible. Lorsqu'une mappe ObjectGrid est configurée avec la stratégie de verrouillage pessimiste, un verrou de transaction pessimiste est obtenu pour une entrée de mappe par la première transaction qui obtient cette entrée à partir de la mappe de sauvegarde. Le verrou pessimiste est maintenu jusqu'à la fin de la transaction. La stratégie de verrouillage pessimiste est généralement utilisée dans les cas suivants :

- Lorsque la mappe de sauvegarde est configurée et que les informations de gestion des versions ne sont pas disponibles.

  Restriction : Les mappes de sauvegarde configurées avec un plug-in de chargeur peuvent lire dans la mappe lors d'une transaction multipartition, mais pas y écrire.

- Lorsque la mappe de sauvegarde est utilisée directement par une application qui nécessite l'assistance de WebSphere eXtreme Scale Client pour le contrôle des accès simultanés.
- Lorsque les informations de gestion des versions sont disponibles mais que les transactions de mise à jour entrent régulièrement en conflit avec les entrées de sauvegarde, ce qui entraîne des échecs de mise à jour optimiste.

La stratégie de verrouillage pessimiste a une incidence considérable sur les performances et l'évolutivité. Par conséquent, réservez son utilisation aux opérations de lecture et d'écriture lorsqu'aucune autre stratégie de verrouillage n'est viable. Par exemple, en cas d'échecs réguliers des mises à jour optimistes ou de reprise après échec optimiste difficile à gérer pour une application.

Lorsque vous utilisez le verrouillage pessimiste, vous pouvez utiliser les méthodes de verrouillage pour

verrouiller des données ou des clés sans renvoyer de valeurs de données. Pour connaître la liste de ces méthodes et les types de verrou qu'elles obtiennent, voir Types de verrou.

Java

Verrouillage optimiste

La stratégie de verrouillage par défaut est OPTIMISTIC. Utilisez le verrouillage optimiste lorsque les données sont rarement modifiées. Les verrous ne sont maintenus que pour un laps de temps limité, c'est-à-dire pendant que les données sont lues à partir du cache et copiées dans la transaction. Lorsque le cache de transaction est synchronisé avec le cache principal, tous les objets mis en cache qui ont été mis à jour sont comparés à la version d'origine. Si la comparaison échoue, la transaction est annulée et une exception OptimisticCollisionException est provoquée.

La stratégie de verrouillage optimiste présuppose qu'il n'est pas possible que deux transactions tentent d'actualiser la même entrée de mappe au même moment. Le verrou n'est pas maintenu pendant toute la durée de la transaction car il est peu probable qu'une autre transaction puisse mettre à jour simultanément la même entrée de mappe. La stratégie de verrouillage optimiste est généralement utilisée dans les situations suivantes :

- Lorsqu'une mappe de sauvegarde est configurée et que les informations sur les versions sont disponibles.

  Restriction : Les mappes de sauvegarde configurées avec un plug-in de chargeur peuvent lire dans la mappe lors d'une transaction multipartition, mais pas y écrire.

- Lorsqu'une mappe de sauvegarde contient essentiellement des transactions qui exécutent des opérations de lecture. Les opérations insert, update ou remove sur les entrées de mappe ne sont pas fréquentes pour la mappe de sauvegarde.
- Lorsqu'une mappe de sauvegarde est insérée, mise à jour ou supprimée plus fréquemment qu'elle n'est lue mais que les transactions entrent rarement en conflit pour la même entrée de mappe.

A l'instar de la stratégie de verrouillage pessimiste, les méthodes de l'interface ObjectMap déterminent comment WebSphere eXtreme Scale Client tente automatiquement d'obtenir un mode de verrouillage pour l'entrée de mappe qui fait l'objet d'un accès. Toutefois, les stratégies pessimiste et optimiste diffèrent sur les points suivants :

- Comme dans la stratégie de verrouillage pessimiste, un mode de verrouillage S est obtenu par les méthodes get et getAll lorsque la méthode est appelée. En revanche, avec le verrouillage optimiste, le mode de verrouillage S n'est pas maintenu jusqu'à la fin de la transaction. Il est libéré avant que la méthode ne rende le contrôle à l'application. L'objectif de l'obtention d'un mode de verrouillage est que WebSphere eXtreme Scale Client vérifie que seules les données validées provenant d'autres transactions soient visibles pour la transaction en cours. Lorsque WebSphere eXtreme Scale Client a vérifié que les données ont été validées, le mode de verrouillage S est libéré. Au moment de la validation, une vérification optimiste des versions est effectuée pour s'assurer qu'aucune autre transaction n'a modifié l'entrée de mappe après la libération du mode de verrouillage S par la transaction en cours. Si une entrée n'est pas extraite de la mappe avant sa mise à jour, son invalidation ou sa suppression, le module d'exécution de WebSphere eXtreme Scale Client extrait implicitement l'entrée de la mappe. Cette opération get implicite est effectuée pour obtenir la valeur en cours au moment de la demande de modification de l'entrée.
- Contrairement à la stratégie de verrouillage pessimiste, les méthodes getForUpdate et getAllForUpdate sont traitées exactement comme les méthodes get et getAll de la stratégie de verrouillage optimiste. Un mode de verrouillage S est obtenu au début de la méthode et est libéré avant le renvoi du contrôle à l'application.

Toutes les autres méthodes ObjectMap sont traitées exactement comme dans le cas de la stratégie de verrouillage pessimiste. Lorsque la méthode commit est appelée, un mode de verrouillage X est obtenu pour toutes les entrées de mappe insérées, mises à jour, supprimées, corrigées ou invalidées, et le mode de verrouillage X est maintenu jusqu'à ce que la transaction termine le traitement de la validation.

La stratégie de verrouillage optimiste considère qu'aucune transaction simultanée ne tente de mettre à jour la même entrée de mappe. En partant de ce principe, il n'est pas nécessaire de maintenir le mode de verrouillage pendant toute la durée de la transaction, car il est peu probable que plusieurs transactions puissent mettre à jour simultanément la même entrée de mappe. Toutefois, étant donné qu'aucun mode de verrouillage n'est maintenu, une autre transaction simultanée peut potentiellement mettre à jour l'entrée de mappe après la libération du mode de verrou S par la transaction en cours.

Pour parer à cette éventualité, WebSphere eXtreme Scale Client obtient un verrou X lors de la validation et effectue une vérification optimiste des versions pour s'assurer qu'aucune autre transaction n'a modifié l'entrée de mappe après que la transaction en cours l'a lue dans la mappe de sauvegarde. Si une autre transaction modifie l'entrée de mappe, la vérification des versions échoue et une exception OptimisticCollisionException se produit. Cette exception force l'annulation de la transaction en cours et

l'application doit retenter la transaction tout entière. La stratégie de verrouillage optimiste s'avère utile lors qu'une mappe est principalement lue et que des mises à jour de la même entrée de mappe sont peu probables.

Java

Aucun verrouillage

Si le verrouillage n'est pas obligatoire car les données ne sont jamais mises à jour ou le sont au cours de période calmes, vous pouvez le désactiver en utilisant la stratégie de verrouillage NONE. Cette stratégie est très rapide car aucun gestionnaire de verrou n'est requis. Elle est idéale pour les tables de recherche et les mappes en lecture seule.

Lorsqu'une mappe de sauvegarde est configurée pour n'utiliser aucune stratégie de verrouillage, aucun verrou de transaction n'est obtenu pour une entrée de mappe.

Restriction : Les mappes de sauvegarde configurées pour utiliser une stratégie de non-verrouillage ne peuvent pas participer à une transaction multipartition.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Types de verrou

Lorsque vous utilisez les verrouillages pessimiste et optimiste, des verrous partagés (S), pouvant être mis à niveau (U) et exclusifs (X) sont utilisés pour maintenir la cohérence. Les verrous optimistes n'étant pas maintenus, c'est lorsque vous configurez le verrouillage pessimiste que les effets du verrouillage sont le plus apparents. Les verrous ont des cycles de vie. Les différents types de verrou ne sont pas tous compatibles entre eux de la même manière. Les verrous doivent être traités dans l'ordre approprié pour éviter les situations d'interblocage.

Verrous partagés, pouvant être mis à niveau et exclusifs

Lorsqu'une application appelle une méthode de l'interface de programmation de mappe, WebSphere eXtreme Scale Client tente automatiquement d'obtenir un verrou pour l'entrée de mappe qui fait l'objet d'un accès.

Java Dans le cas des applications Java, les verrous sont également obtenus lorsqu'une application utilise une méthode de recherche sur un index ou lance une requête.

Lorsque vous utilisez le verrouillage pessimiste, vous pouvez utiliser les méthodes de verrouillage pour verrouiller des données ou des clés sans renvoyer de valeurs de données. Ces méthodes vous permettent de verrouiller la clé dans la grille, ou de verrouiller la clé et de déterminer si la valeur existe dans la grille.

LockMode est une énumération qui vous permet d'indiquer les clés que vous souhaitez verrouiller à l'aide des valeurs possibles suivantes :

- Java Partagé (SHARED), pouvant être mis à niveau (UPGRADEABLE) et exclusif (EXCLUSIVE)
- .NET Partagé (Shared), pouvant être mis à niveau (Upgradeable) et exclusif (Exclusive)

WebSphere eXtreme Scale Client utilise les modes de verrouillage suivants en fonction de la méthode que l'application appelle dans l'interface de programmation de mappe :

Verrouillage S

Mode de verrouillage partagé pour la clé d'une entrée de mappe. La durée pendant laquelle le verrou S est maintenu dépend du niveau d'isolement de transaction utilisé. Avec le mode de verrouillage S, les transactions qui tentent d'obtenir un verrou S ou un verrou U (pouvant être mis à niveau) sur une même clé bénéficient d'un accès simultané, tandis que les autres transactions qui tentent d'obtenir un verrou X (exclusif) sur cette même clé sont bloquées.

Verrouillage U

Mode de verrouillage pouvant être mis à niveau pour la clé d'une entrée de mappe. Le verrou U est maintenu jusqu'à la fin de la transaction. Avec le mode de verrouillage U, les transactions qui obtiennent un verrou S sur une même clé bénéficient d'un accès simultané, tandis que les autres transactions qui tentent d'obtenir un verrou U ou un verrou X sur cette même clé sont bloquées.

Verrouillage X

Mode de verrouillage exclusif pour la clé d'une entrée de mappe. Le verrou X est maintenu jusqu'à la fin de la transaction. Avec le mode de verrouillage X, une seule transaction peut insérer, mettre à jour ou supprimer une entrée de mappe d'une valeur de clé donnée. Un verrou X bloque toutes les autres transactions qui tentent d'obtenir un verrou S, U ou X sur cette même clé.

Le mode de verrouillage S est plus faible que le mode de verrouillage U car il permet à davantage de transactions d'accéder simultanément à la même entrée de mappe. Le mode de verrouillage U est légèrement plus fort que le mode de verrouillage S car il bloque les autres transactions qui demandent un mode de verrouillage U ou X. Le mode de verrouillage S bloque uniquement les autres transactions qui demandent un mode de verrouillage X. Cette petite différence est pourtant importante lorsqu'il s'agit d'empêcher l'occurrence de certains interblocages. Le mode de verrouillage X est le plus fort car il bloque toutes les autres transactions qui tentent d'obtenir un verrou S, U ou X sur l'entrée de mappe concernée. Il garantit qu'une seule transaction peut insérer, mettre à jour ou supprimer une entrée de mappe, et empêche la perte des mise à jour lorsque plusieurs transactions tentent de mettre à jour la même entrée.

Consultez le tableau ci-dessous pour comprendre la relation entre ces modes de verrouillage et le comportement des méthodes existantes :

Tableau 1. Modes de verrouillage et méthodes équivalentes

Modes de verrouillage (Java / .NET)	Méthode Java équivalente	Méthode .NET équivalente
SHARED / Shared (partagé)	Méthodes get et getAll de l'interface ObjectMap, méthodes d'index et requêtes.	Méthodes Get(), GetAndLock(Key,LockMode. Shared), GetAndLockAll(KeyList, LockMode.Shared), GetAll.

UPGRADABLE / Upgradable (pouvant être mis à niveau)	getForUpdate(), getAllForUpdate()	GetAndLock(Key, LockMode.Updgradable), GetAndLockAll(KeyList, LockMode.Upgradable)
EXCLUSIVE / Exclusive (exclusif)	Méthodes getNextKey(), commit(), put, putAll, remove, removeAll, insert, update et touch ; méthodes invalidate et invalidateAll globales. (Aucun verrou n'est obtenu pour les méthodes invalidate et invalidateAll locales, car aucune des entrées de la mappe de sauvegarde (BackingMap) n'est invalidée par les appels à la méthode d'invalidation locale.) Java **Remarque :** Les méthodes upsert et upsertAll remplacent les méthodes put et putAll d'ObjectMap. Utilisez la méthode upsert pour indiquer à la mappe de sauvegarde et au chargeur qu'une entrée dans la grille de données doit placer la clé et la valeur dans la grille. La mappe de sauvegarde et le chargeur exécutent une insertion ou une mise à jour pour placer la valeur dans la grille et le chargeur. Si vous exécutez l'API upsert dans vos applications, le chargeur reçoit un type LogElement UPSERT, ce qui permet aux chargeurs d'exécuter des appels de fusion (merge) de base de données ou upsert au lieu d'insert ou update.	Commit(), Add, AddAll, Put, PutAll, Remove, RemoveAll, Replace, ReplaceAll, Touch, TouchAll, Invalidate, InvalidateAll **Remarque :** Les méthodes Put et PutAll sont équivalentes aux méthodes Java upsert et upsertAll.

Le tableau ci-dessous est une matrice de compatibilité des modes de verrouillage, qui résume les modes de verrouillage décrits et permet de déterminer la compatibilité entre les différents modes. Comment lire cette matrice : la première colonne indique le mode de verrouillage déjà octroyé. Pour chaque colonne suivante, l'en-tête indique le mode de verrouillage demandé par une autre transaction. Si une case contient `Oui`, le mode de verrouillage demandé par l'autre transaction est octroyé car il est compatible avec le mode de verrouillage déjà octroyé. Si une case contient `Non`, le mode de verrouillage demandé n'est pas octroyé, car il n'est pas compatible. Dans ce cas, l'autre transaction doit attendre que la première transaction libère le verrou qu'elle détient.

Tableau 2. Matrice de compatibilité des modes verrouillage

Verrou	**Verrou S (partagé)**	**Verrou U (pouvant être mis à niveau)**	**Verrou X (exclusif)**	**Force**
S (partagé)	Oui	Oui	Non	le plus faible
U (pouvant être mis à niveau)	Oui	Non	Non	normal
X (exclusif)	Non	Non	Non	le plus fort

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Interblocages

Un interblocage peut se produire lorsque deux transactions tentent de mettre à jour la même entrée de cache.

Exemple classique d'interblocage

Examinez la séquence de demandes de mode de verrouillage suivante :

1. Verrou X octroyé à la transaction 1 pour clé1.
2. Verrou X octroyé à la transaction 2 pour clé2.
3. Verrou X demandé par la transaction 1 pour clé2. (La transaction 1 est bloquée dans l'attente du verrou détenu par la transaction 2.)
4. Verrou X demandé par la transaction 2 pour clé1. (La transaction 2 est bloquée dans l'attente du verrou détenu par la transaction 1.)

La séquence précédente est l'exemple type de l'interblocage de deux transactions qui tentent d'acquérir plusieurs verrous dans un ordre différent. Pour éviter cet interblocage, chaque transaction doit obtenir ces verrous dans le même ordre.

Java

Prévention des interblocages grâce au verrouillage optimiste

Si la stratégie de verrouillage OPTIMISTE est utilisée et que la méthode flush sur l'interface ObjectMap n'est jamais utilisée par l'application, les modes de verrouillage ne sont demandés par la transaction que lors du cycle de validation. Pendant le cycle de validation, WebSphere eXtreme Scale Client détermine les clés pour les entrées de mappe qui doivent être verrouillées et demande les modes de verrouillage dans l'ordre des clés (comportement déterministe). Avec cette méthode, WebSphere eXtreme Scale Client évite la plupart des interblocages classiques.

Toutefois, eXtreme Scale n'empêche pas et ne peut pas empêcher tous les interblocages. Quelques scénarios doivent être gérés par l'application. Voici les cas dont l'application doit tenir compte et dans lesquels elle doit prendre des mesures préventives :

Il existe un cas dans lequel WebSphere eXtreme Scale Client est capable de détecter un interblocage sans être obligé d'attendre l'expiration du délai d'attente du verrou. Si ce cas se produit, une exception com.ibm.websphere.objectgrid.LockDeadlockExceptionexception est émise. Examinez cet exemple de code :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
sess.begin();
Person p = (IPerson)person.get("Lynn");
// Lynn had a birthday; so make her 1 year older.
p.setAge( p.getAge() + 1 );
person.put( "Lynn", p );
sess.commit();
```

Dans le même scénario, vous pouvez utiliser la méthode upsert dans l'exemple de code suivant :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
sess.begin();
Person p = (IPerson)person.get("Lynn");
// Lynn had a birthday; so make her 1 year older.
p.setAge( p.getAge() + 1 );
person.upsert( "Lynn", p );
sess.commit();
```

Dans cette situation, les deux transactions tentent de mettre à jour l'âge de l'objet personne Lynn. Ces deux transactions possèdent un verrou S sur l'entrée Lynn de la mappe PERSON suite à l'appel de la méthode person.get("Lynn"). Suite à l'appel de la méthode person.put ("Lynn", p), ces deux transactions tentent de transformer le verrou S en verrou X. Elles sont donc toutes les deux bloquées en attendant que l'autre transaction libère le verrou S qu'elle détient. Par conséquent, un interblocage se produit car un état d'attente circulaire existe entre les deux transactions. Un état d'attente circulaire existe lorsque plusieurs transactions tentent de transformer un verrou faible en verrou fort pour la même entrée de mappe. Dans ce scénario, une exception LockDeadlockException est émise au lieu d'une exception LockTimeoutException.

Dans les applications Java, l'application peut empêcher l'émission de l'exception LockDeadlockException dans l'exemple précédent en utilisant la stratégie de verrouillage optimiste au lieu de la stratégie de verrouillage pessimiste. La stratégie de verrouillage optimiste constitue la solution privilégiée lorsque la

mappe est essentiellement lue et que les mises à jour ne sont pas fréquentes.

Java .NET

Prévention des interblocages grâce au verrouillage pessimiste

.NET **Avertissement :** Les applications .NET ne prennent en charge que le verrouillage pessimiste. La suite de cette rubrique fait référence aux noms de méthode Java. Toutefois, les explications sont également valables pour les noms de méthode .NET. Ces méthodes sont les suivantes : Get, GetAndLock, GetAndLockAll, Put, Add, Replace et Remove.

Pour éviter les interblocages lorsque vous utilisez la stratégie de verrouillage pessimiste, procédez comme suit :

- Utilisez le niveau d'isolement de transaction `READ_COMMITTED` (lecture validée). Ce niveau empêche le verrou S acquis par la méthode get d'être maintenu jusqu'à la fin de la transaction. Si la clé n'est jamais invalidée dans le cache transactionnel, les lectures reproductibles sont toujours garanties.
- Utilisez d'autres méthodes d'obtention que la méthode get.
  - Java Utilisez la méthode getForUpdate.
  - .NET Utilisez la méthode GetAndLock ou GetAndLockAll.

  La première transaction qui appelle la méthode getForUpdate obtient un verrou U et non un verrou S. Avec ce mode de verrouillage, la deuxième transaction est bloquée lorsqu'elle appelle la méthode getForUpdate. En effet, le verrou U n'est octroyé qu'à une seule transaction. La deuxième transaction étant bloquée, elle ne possède aucun verrou sur l'entré de mappe. La première transaction n'est pas bloquée lorsqu'elle tente de mettre à niveau le mode de verrouillage U vers le mode de verrouillage X suite à l'appel de la méthode put. Voilà pourquoi on dit du mode de verrouillage U qu'il peut être mis à niveau. Lorsque la première transaction est terminée, la deuxième est débloquée et le verrou U lui est octroyé. Pour empêcher ce scénario d'interblocage de la mise à niveau du mode de verrouillage, une application peut employer la méthode getForUpdate au lieu de la méthode get en cas d'utilisation de la stratégie de verrouillage pessimiste.

  Important : Cette solution n'empêche pas les transactions en lecture seule de lire une entrée de mappe. Ces transactions appellent la méthode get, mais jamais les méthodes put, insert, update et remove. Les accès simultanés sont aussi nombreux que lors de l'utilisation de la méthode get classique. Ils ne sont réduits que lorsque la méthode getForUpdate est appelée par plusieurs transactions pour la même entrée de mappe.

  Lorsque des transactions appellent la méthode getForUpdate pour plusieurs entrées de mappe, vous devez le savoir pour être sûr que les verrous U sont acquis dans le même ordre par chaque transaction. Par exemple, supposons que la première transaction appelle la méthode pour la clé 1 et la clé 2, et qu'une autre transaction simultanée appelle la méthode pour les mêmes clés mais dans l'ordre inverse. Cette séquence entraîne l'interblocage classique car plusieurs verrous sont obtenus dans un ordre différent par des transactions différentes. Afin d'éviter tout interblocage, l'application doit toujours s'assurer que chaque transaction accède à plusieurs entrées de mappe dans l'ordre des clés. Etant donné que le verrou U est obtenu lors de l'appel de la méthode getForUpdate et non au moment de la validation, WebSphere eXtreme Scale Client ne peut pas ordonner les demandes de verrouillage de la même manière que lors du cycle de validation. Dans ce cas, c'est l'application qui doit contrôler l'ordre de verrouillage.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Accès aux données et transactions

WebSphere eXtreme Scale Client utilise des transactions. Dès lors qu'une application dispose d'une connexion à une grille de données, vous pouvez accéder aux données et interagir avec celles-ci dans la grille.

Java

Transactions dans les applications Java

Avec les applications Java, vous pouvez établir une connexion client avec une instance répartie.

Lorsqu'une application interagit avec une Session, elle doit se trouver dans le contexte d'une transaction. Une transaction est initiée, puis validée ou annulée, en appelant les méthodes Session.begin, Session.commit et Session.rollback sur l'objet Session. Les applications peuvent également utiliser le mode de validation automatique, dans lequel la Session initie et valide automatiquement une transaction lorsque l'application interagit avec des mappes. Le mode de validation automatique est toutefois plus lent.

Dans une application Java, chaque unité d'exécution doit avoir sa propre Session. Lorsque vous souhaitez que l'application utilise ObjectGrid sur une unité d'exécution, appelez l'une des méthodes getSession pour obtenir une session. Lorsque l'application n'utilise plus la session, utilisez la méthode Session.close(). Cette méthode ferme la session, la renvoie au pool et libère ses ressources. La fermeture d'une session est facultative, mais elle améliore les performances des appels suivants vers la méthode getSession(). Si l'application utilise une infrastructure d'injection de dépendance telle que Spring, vous pouvez injecter une Session dans le bean d'une application lorsque cela est nécessaire.

Lorsque vous avez obtenu une Session, l'application peut accéder aux données stockées dans des mappes de la grille d'objets. L'API fondée sur les mappes s'obtient à l'aide de la méthode Session.getMap.

.NET

Transactions dans les applications .NET

Dans les applications .NET, chaque unité d'exécution doit posséder son propre objet IGridMapPessimisticTx ou IGridMapPessimisticAutoTx. Avec l'objet IGridMapPessimisticTx, vous utilisez la propriété Transaction pour commencer, valider ou annuler explicitement la transaction. Avec l'objet IGridMapPessimisticAutoTx, ces trois opérations ont lieu automatiquement. Dès que vous avez obtenu l'un de ces objets, l'application peut accéder aux données stockées dans la grille de données.

Logique d'utilisation des transactions

Les transactions peuvent paraître lentes, mais vous devez les utiliser pour les raisons suivantes :

1. Pour pouvoir annuler des modifications en cas d'exception ou si la logique métier requiert l'annulation d'une modification d'état.
2. Pour placer des verrous sur les données et les libérer tout au long de la durée de vie d'une transaction, afin que les modifications soient apportées de manière atomique (toutes les modifications sont appliquées, ou aucune).
3. Pour générer une unité atomique de réplication.
4. Java Pour mettre à jour plusieurs partitions.

Vous pouvez personnaliser le niveau de prise en charge des transactions nécessaire. Votre application peut désactiver la prise en charge de l'annulation et le verrouillage, mais cela sera à ses dépens. En effet, elle devra alors gérer elle-même l'absence de ces fonctions. Voici des exemples de la manière dont une application peut gérer la prise en charge des transactions :

- Java Une application peut désactiver le verrouillage en configurant la stratégie de verrouillage NONE pour la mappe dynamique. Pour plus d'informations, voir Création de mappes dynamiques avec les API Java. Cette stratégie permet de gagner en rapidité, mais plusieurs transactions simultanées peuvent désormais modifier les mêmes données sans protection les unes des autres. L'application est responsable du verrouillage et de la cohérence des données lorsque NONE est utilisé. Cette option n'est pas valide pour les applications WebSphere eXtreme Scale Client for .NET, qui ne prennent en charge que la stratégie de verrouillage PESSIMISTIC.

Java Si l'application modifie un objet récupéré à l'aide de la valeur CopyMode NONE, elle modifie directement la copie validée de cet objet. Dans ce mode, l'annulation de la transaction n'a aucune signification. Vous modifiez la seule copie dans l'ObjectGrid. Bien que l'utilisation de CopyMode NONE offre une grande rapidité, vous devez en mesurer les conséquences. Une application utilisant ce mode ne doit jamais annuler la transaction. Si elle le fait, les modifications ne sont pas reportées dans l'index *et* ne sont pas répliquées si la réplication est activée.

Les valeurs par défaut sont plus simples à utiliser et risquent moins d'entraîner des d'erreurs. Si vous favorisez les performances au détriment de la fiabilité des données, l'application doit savoir ce qu'elle fait afin d'éviter tout problème.

Transactions et partitions

Les transactions des applications Java peuvent mettre à jour une ou plusieurs partitions, mais par défaut une seule partition est mise à jour. Les applications .NET ne peuvent mettre à jour qu'une seule partition.

Utilisez l'API de session TxCommitProtocol pour activer le support de transaction multipartition pour WebSphere eXtreme Scale Client. Vous pouvez utiliser les deux options suivantes :

- TxCommitProtocol.ONEPHASE (par défaut) : les transactions d'un client peuvent lire plusieurs partitions mais ne peuvent en mettre à jour qu'une seule. Les tentatives de mise à jour de plusieurs partitions échouent.
- TxCommitProtocol.TWOPHASE : les transactions d'un client peuvent lire et mettre à jour plusieurs partitions. Elles utilisent le protocole de validation en deux phases pour que les données écrites dans les partitions soient automatiquement validées ou annulées. Si une transaction écrit dans une seule partition, le protocole de validation en une phase est utilisé.

Rubrique parent : Présentation du traitement des transactions

Tâches associées:

Java Développement d'applications pour écrire dans des transactions multipartitions pour WebSphere eXtreme Scale dans un environnement autonome

Java Interaction avec les données dans une transaction pour les applications Java

.NET Interaction avec les données dans une transaction pour les applications .NET

Isolement des transactions

Vous pouvez utiliser l'un des trois niveaux d'isolement de transaction suivants afin d'optimiser la sémantique de verrouillage qui maintient la cohérence dans chaque mappe de cache : lecture reproductible, lecture validée et lecture non validée.

Présentation de l'isolement de transaction

L'isolement de transaction définit comment les modifications apportées par une opération deviennent visibles aux autres opérations simultanées.

Vous pouvez définir l'un des trois niveaux d'isolement de transaction suivants afin d'optimiser la sémantique de verrouillage que WebSphere eXtreme Scale Client utilise pour maintenir la cohérence dans chaque mappe de cache : lecture reproductible, lecture validée et lecture non validée.

Vous pouvez définir le niveau d'isolement de transaction de l'une des manières suivantes :

- Java Dans l'interface Session avec la méthode setTransactionIsolation. L'isolement de transaction peut être modifié à tout moment pendant la durée de vie de la session si une transaction n'est pas en cours d'exécution.
- .NET Dans l'interface IGridTransaction avec la propriété TransactionIsolationLevel.

Le produit f=garantit le respect des divers aspects de la sémantique d'isolement de transaction en paramétrant la demande et le maintien des verrous (S) partagés. L'isolement de transaction n'a aucune incidence sur les mappes configurées pour utiliser la stratégie de verrouillage optimiste ou aucune stratégie de verrouillage, ou lorsque des verrous pouvant être mis à jour (U) sont obtenus.

Lecture reproductible avec verrouillage pessimiste

Le niveau d'isolement de transaction Lecture reproductible est le niveau par défaut. Ce niveau d'isolement empêche les lectures de pages modifiées et les lectures non reproductibles mais pas les lectures fantômes. Une lecture de pages modifiées est une opération de lecture de données qui ont été modifiées par une transaction mais qui n'ont pas été validées. Une lecture non reproductible peut survenir lorsqu'aucun verrou en lecture n'a été obtenu lors de l'opération de lecture. Une lecture fantôme a lieu lorsque deux opérations de lecture identiques ont été effectuées mais que deux jeux de résultats ont été renvoyés en raison d'une mise à jour intervenue entre les opérations de lecture. Dans les applications Java, les lectures fantômes sont possibles lorsque vous utilisez des requêtes ou des index, car les verrous ne sont pas obtenus pour des plages de données mais uniquement pour les entrées de cache qui correspondent à l'index ou aux critères de requête. Le produit obtient une lecture reproductible en maintenant les verrous S jusqu'à la fin de la transaction qui détient le verrou concerné. Etant donné qu'un verrou X n'est octroyé que lorsque tous les verrous S sont libérés, toutes les transactions maintenant un verrou S obtiendront obligatoirement la même valeur lors de la relecture.

Lecture validée avec verrouillage pessimiste

Le niveau d'isolement de transaction de lecture validée peut être utilisé avec WebSphere eXtreme Scale Client, ce qui empêche les lectures de pages modifiées mais pas les lectures non reproductibles ni les lectures fantômes ; WebSphere eXtreme Scale Client continue ainsi à utiliser des verrous S pour lire les données de la mappe de cache mais libère immédiatement ces verrous.

Lecture non validée avec verrouillage pessimiste

Le niveau d'isolement de transaction lecture non validée peut être utilisé avec WebSphere eXtreme Scale Client, ce qui autorise les lectures de pages modifiées, les lectures non reproductibles et les lectures fantômes.

Rubrique parent : Présentation du traitement des transactions

Référence associée:
Java Exemples Java pour l'isolement de transaction

Java

Validation en deux phases et reprise sur incident

2.5+ Le protocole de validation en deux phases coordonne toutes les partitions qui participent à une transaction répartie, en déterminant si la transaction doit être validée ou annulée.

Dans une grille de données répartie, les partitions sont réparties sur plusieurs machines virtuelles Java™. Ces machines JVM peuvent se trouver sur plusieurs systèmes. Une transaction qui écrit dans plusieurs partitions peut impliquer des décisions transactionnelles qui ont une incidence sur plusieurs systèmes. Lorsque la transaction est validée à l'aide d'un protocole de validation en deux phases, ce processus de validation conserve l'ensemble de la transaction ou ne la conserve pas du tout. Le processus de validation en deux phases garantit ce résultat même en présence d'erreurs liées au système, aux partitions ou à la communication. Si un incident survient au cours de la deuxième phase, le client WebSphere eXtreme Scale tente de résoudre le problème automatiquement, sauf si l'erreur répond à certains critères pour lesquels vous pouvez intervenir manuellement.

Une transaction prévue pour écrire dans plusieurs partitions utilise le protocole de validation en deux phases. Ce protocole garantit que le processus de validation est cohérent dans tous les partitions et systèmes. WebSphere eXtreme Scale fait office de coordinateur qui contrôle le processus de validation en deux phases. Les partitions impliquées dans la transaction sont appelées participants ou gestionnaires de ressources (RM). Au cours de la deuxième phase du protocole de validation, le coordinateur délègue l'une des partitions pour faire office de gestionnaire de transactions (TM). Le gestionnaire de transactions assure le suivi de la décision concernant chaque transaction et la reprise de la transaction en cas de problème.

Première phase :

Lorsqu'une application valide une transaction, le client WebSphere eXtreme Scale démarre la première phase en envoyant une demande de préparation de validation à chaque partition identifiée comme gestionnaire de ressources. Chaque partition applique les modifications de transaction aux mappes de sauvegarde et conserve tous les verrous pour protéger l'intégrité des données. Le gestionnaire de ressources notifie le client WebSphere eXtreme Scale. Une fois que toutes les partitions identifiées comme gestionnaire de ressources ont répondu avec succès, le client WebSphere eXtreme Scale lance la seconde phase du protocole de validation.

Seconde phase :

Si une ou plusieurs partitions échouent au cours de la première étape, le coordinateur annule (invalide) toutes les partitions au cours de la deuxième phase. En revanche, si toutes les partitions RM répondent avec succès, le client WebSphere eXtreme Scale délègue l'une des partitions pour faire office de partition TM. En tant que coordinateur, WebSphere eXtreme Scale lance la seconde phase du protocole de validation en envoyant une demande de validation ou d'annulation à toutes les partitions impliquées dans la transaction. Chaque partition identifiée comme RM applique les modifications à la mappe de sauvegarde ou les annule, puis lève tous les verrous. Le gestionnaire de ressources RM notifie ensuite le client WebSphere eXtreme Scale. Si une ou plusieurs partitions ont échoué au cours de la seconde phase, la partition TM déléguée reprend automatiquement la transaction. Cette reprise automatique garantit que toutes les partitions impliquées dans la transaction sont cohérentes.

Phase d'attente de validation :

La phase d'attente de validation est la période entre le traitement réussi de la première phase par la partition RM et le début de la seconde phase. Pendant cette période, la partition RM ne sait pas si elle doit valider ou annuler la transaction. La partition RM maintient les verrous. Dans ce cas, les autres transactions peuvent être affectées par une augmentation des conflits de verrouillage.

Reprise après incident au cours d'une validation en deux phases

En cas d'erreur au cours de la première phase, le client WebSphere eXtreme Scale annule la transaction. Si l'une des partitions ne parvient pas à valider la transaction, le gestionnaire de transactions retente régulièrement de valider la transaction. Voici un exemple des messages qui sont consignés dans le journal dans ce scénario :

```
00000099 TransactionLog I CWOBJ8705I: La résolution automatique de la transaction WXS-
40000139-DF01-216D-E002-1CB456931719 sur RM:TestGrid:TestSet2:20 attend toujours une
décision. Une nouvelle tentative de résolution de la transaction sera exécutée dans 30
secondes.
```

Laissez le client WebSphere eXtreme Scale résoudre la transaction. N'intervenez manuellement que si la transaction n'est pas récupérée après une minute ou que l'application est affectée par un grand nombre de conflits de verrous parce qu'il s'agit d'une transaction en attente. Pour plus d'informations sur la reprise manuelle d'une transaction, voir Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition.

Rubrique parent : Présentation du traitement des transactions

Rubrique parent : 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Concepts associés:
Java Stratégies de verrouillage

Tâches associées:
Java Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition
Java Résolution des exceptions de délai d'attente de verrouillage

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7 Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE "EN L'ETAT" SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM® Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des

logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances, ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Tous les tarifs indiqués sont les prix de vente actuels suggérés par IBM et sont susceptibles d'être modifiés sans préavis. Les tarifs appliqués peuvent varier selon les revendeurs.

Ces informations sont fournies uniquement à titre de planification. Elles sont susceptibles d'être modifiées avant la mise à disposition des produits décrits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les exemples de programmes sont fournis "en l'état", sans garantie d'aucune sorte. IBM ne sera en aucun cas responsable de tout dommage résultant de l'utilisation des exemples de programmes.

Toute copie totale ou partielle de ces exemples de programmes et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit :

© nom de votre société) (année). Des segments de code sont dérivés des exemples de programmes d'IBM Corp.

© Copyright IBM Corp. _entrez l'année ou les années_. All rights reserved.

Documentation sur l'interface de programmation

Cette publication contient principalement des informations qui ne sont destinées à être utilisées comme interfaces de programmation de WebSphere eXtreme Scale. Il décrit également des interfaces de programmation qui permettent au Client d'écrire des programmes pouvant utiliser les services de WebSphere eXtreme Scale. Ces informations sont identifiées par un texte d'introduction dans un chapitre ou une section ou par le repère suivant : Informations relatives aux interfaces de programmation.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à l'adresse www.ibm.com/legal/copytrade.shtml.

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Remarques sur les règles de confidentialité

Les produits IBM® Software ("Offres logicielles"), dont les solutions de type SaaS (logiciel sous forme de services), peuvent utiliser des cookies ou d'autres technologies pour collecter des informations sur l'utilisation des produits, afin notamment d'améliorer l'expérience des utilisateurs finaux ou encore de personnaliser les interactions avec ces derniers. Dans de nombreux cas, ces Offres logicielles ne collectent pas d'informations permettant d'identifier l'utilisateur. Certaines d'entre elles peuvent toutefois vous aider à collecter ce type d'informations. Si la présente Offre logicielle utilise des cookies pour collecter des informations identifiant la personne, des informations spécifiques sur l'utilisation des cookies par cette offre figurent ci-après.

Cette Offre logicielle n'utilise pas de cookies ni d'autre technologie pour collecter des informations permettant d'identifier l'utilisateur.

Si les configurations déployées pour cette Offre logicielle vous permettent en tant que client de collecter des informations identifiant la personne auprès des utilisateurs finaux, via des cookies et d'autres technologies, il vous est conseillé de vous renseigner auprès de votre conseil juridique sur les lois applicables à cette collecte de données et notamment aux exigences en matière de notification et d'accord.

Pour plus d'informations concernant l'utilisation de différentes technologies, et notamment des cookies, à ces fins, consultez la politique IBM de protection des renseignements personnels (http://www.ibm.com/privacy) et la déclaration Online Privacy Statement d'IBM (http://www.ibm.com/privacy/details/us/en), et particulièrement les sections intitulées "Cookies, Web Beacons and Other Technologies" et "Software Products and Software-as-a Service".

Rubrique parent : Présentation générale d'IBM WebSphere DataPower XC10 Appliance

Scénarios

Le scénario utilise des informations du monde réel pour construire une image complète. Exécutez un scénario pour comprendre nouveaux les concepts ou pour accomplir des tâches communes.

Scénario : Configuration d'une grille de données d'entreprise
Configurez une grille de données d'entreprise lorsque vous voulez connecter des applicationsJava™ et .NET à la même grille de données.

2.5+ **Sécurisation de WebSphere DataPower XC10 Appliance**
Les grilles de données du dispositif stockent des informations qui sont confidentielles et doivent être protégées.

Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Vous pouvez migrer une session de réplication de mémoire à mémoire ou une session de base de données pour utiliser la gestion de session WebSphere eXtreme Scale.

Scénario : Configuration d'une grille de données d'entreprise

Configurez une grille de données d'entreprise lorsque vous voulez connecter des applicationsJava™ et .NET à la même grille de données.

Avant de commencer

- Installez le produit. Pour les clients, vous pouvez utiliser des clients Java et .NET. Pour plus d'informations, voir Installation de WebSphere DataPower XC10 Appliance.
- Si vous effectuez une mise à niveau à partir d'une version précédente, tous les serveurs de conteneur et de catalogue doivent être au même niveau d'édition. Pour plus d'informations, voir Mise à jour de WebSphere DataPower XC10 Appliance.

1. Présentation de la grille de données d'entreprise
   Les grilles de données d'entreprise utilisent le mécanisme de transport eXtremeIO et un nouveau format de sérialisation. Avec le nouveau transport et le nouveau format de sérialisation, vous pouvez connecter des clients Java et .NET à une même grille de données.
2. Configuration d'IBM eXtremeIO (XIO)
   IBM® eXtremeIO (XIO) est un mécanisme de transport qui remplace ORB (Object Request Broker).
3. Développement d'applications de grille de données d'entreprise
   Après avoir configuré IBM eXtremeIO, vous pouvez écrire des applications qui accèdent à la grille de données d'entreprise.

Rubrique parent : 2.5+ Scénarios

Configuration d'IBM eXtremeIO (XIO)

2.5+ IBM® eXtremeIO (XIO) est un mécanisme de transport qui remplace ORB (Object Request Broker).

Avant de commencer

- 2.5 Pour configurer XIO, vous devez posséder WebSphere eXtreme Scale Client version 8.6.0.2 ou ultérieur.

Vous pouvez configurer XIO pour tous les serveurs de conteneur du domaine de service de catalogue en activant XIO dans les serveurs de catalogue. Les serveurs de conteneur reconnaissent le type de transport du serveur de catalogue et utilisent ce type de transport.

2.5

Pourquoi et quand exécuter cette tâche

XIO est le mécanisme de transport par défaut si vous créez une nouvelle configuration avec la version 2.5 ou suivante. Si vous effectuez une mise à niveau depuis une version précédente du microprogramme, votre paramètre de transport est défini pour utiliser ORB.

Procédure

Activez le mécanisme de transport XIO dans la collectivité. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Paramètres** > **Services de communication**. Sélectionnez le mécanisme de transport XIO. Le paramètre de transport est un paramètre de niveau collectivité. Par conséquent, si vous mettez à jour le paramètre de transport, tous les dispositifs de la collectivité doivent être redémarrés.

Résultats

Les serveurs que vous avez configurés utilisent le transport XIO. Pour vérifier que la configuration est correcte, voir Affichage du type de transport du domaine de service de catalogue.

La collectivité utilise le mécanisme de transport XIO. Lorsque vous activez XIO, vous activez également :

eXtreme Data Format (XDF)

XDF sérialise et stocke les clés et les valeurs dans la grille de données dans un format indépendant du langage. Si vous utilisez XDF, les applications Java et .NET peuvent accéder aux mêmes objets de grille de données.

IBM eXtremeMemory

eXtremeMemory vous aide à éviter les pauses de récupération d'espace, ce qui permet stabiliser les performances et de bénéficier de temps de réponse plus prévisible.

2.5+ **Affichage du type de transport du domaine de service de catalogue**
Vous pouvez afficher le type de transport qui est actuellement utilisé pour le domaine de service de catalogue.

Rubrique parent : Scénario : Configuration d'une grille de données d'entreprise
Rubrique précédente : Présentation de la grille de données d'entreprise
Rubrique suivante : Développement d'applications de grille de données d'entreprise
Rubrique parent : Configuration des collectivités et des zones
Rubrique précédente : Configuration d'une réplication multimaître entre les collectivités

Développement d'applications de grille de données d'entreprise

Après avoir configuré IBM eXtremeIO, vous pouvez écrire des applications qui accèdent à la grille de données d'entreprise.

Avant de commencer

- Vous devez déjà disposer d'applications Java ou .NET qui accèdent à la grille de données. Pour plus d'informations sur l'écriture d'applications, voir Module 2 du guide d'initiation : Création d'une application client.

Extension de classe
Le format de données XDF (eXtreme data format) permet d'étendre les classes. Avec l'extension de classe, vous pouvez faire évoluer les définitions de classe utilisées dans la grille de données sans affecter les anciennes applications qui utilisent les version précédentes d'une classe. Ces anciennes classes sont des données d'accès qui se trouvent dans la même mappe que les nouvelles applications.

Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET
Utilisez des annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes Java et .NET.

Mappage de clés avec des partitions avec des annotations PartitionKey
Un alias PartitionKey est utilisé pour identifier les zones ou les attributs sur lesquels un calcul de code de hachage est exécuté pour déterminer la partition dans laquelle les données sont sauvegardées. L'annotation PartitionKey n'est valide que sur les attributs de clé.

Types de données équivalents entre Java et C#
Lorsque vous développez des applications de grille de données d'entreprise, les types de données entre les applications Java et C# doivent être compatibles.

Rubrique parent : Scénario : Configuration d'une grille de données d'entreprise
Rubrique précédente : Configuration d'IBM eXtremeIO (XIO)

Extension de classe

Le format de données XDF (eXtreme data format) permet d'étendre les classes. Avec l'extension de classe, vous pouvez faire évoluer les définitions de classe utilisées dans la grille de données sans affecter les anciennes applications qui utilisent les version précédentes d'une classe. Ces anciennes classes sont des données d'accès qui se trouvent dans la même mappe que les nouvelles applications.

Présentation

L'extension de classe est une autre extension de l'identification des classes et des zones qui détermine si deux types sont suffisamment compatibles pour fonctionner conjointement. Les classes peuvent fonctionner ensemble lorsqu'une des classes dispose d'un nombre de zones inférieur à l'autre classe. Les scénarios utilisateur suivants sont intégrés dans l'implémentation :

Plusieurs versions d'une même classe d'objets

Dans ce scénario, vous disposez d'une mappe dans une application de vente qui permet de suivre les clients. Cette mappe dispose de deux interfaces. La première est dédiée aux achats en ligne et la seconde, aux achats effectués par téléphone. Dans la version 2 de cette application de vente, vous décidez d'accorder une réduction sur les achats en ligne en fonction des habitudes d'achat des clients. Cette réduction est stockée avec l'objet Client. Les employés de la vente par téléphone utilisent toujours la version 1 de l'application qui ne sait pas qu'il existe une nouvelle zone de réduction dans la version en ligne. Vous voulez que les objets Client de la version 2 de l'application fonctionnent avec les objets Client créés avec la version 1 de l'application et vice versa.

Plusieurs versions d'une classe d'objets différente

Dans ce scénario, vous disposez d'une application de vente écrite en Java™ qui conserve une mappe des objets Client. Vous disposez également d'une autre application écrite en C# qui permet de gérer l'inventaire de l'entrepôt et qui expédie les commandes des clients. Ces classes sont actuellement compatibles en fonction des noms des classes, des zones et des types. Dans l'application de vente Java, vous voulez ajouter une option à l'enregistrement Client pour associer le vendeur à un compte de client. Cependant, vous ne voulez pas mettre à jour l'application d'entrepôt pour stocker cette zone, car elle est inutile dans l'entrepôt.

Plusieurs versions incompatibles d'une même classe

Dans ce scénario, les applications de vente et d'inventaire contiennent toutes les deux un objet Client. L'application d'inventaire utilise une zone d'ID qui correspond à une chaîne et l'application de vente, une zone d'ID qui est un entier. Ces types ne sont pas compatibles. Par conséquent, les objets ne sont probablement pas stockés dans la même mappe. Les objets doivent être gérés par la sérialisation XDF et traités comme deux types distincts. Bien que ce scénario n'entre pas réellement dans le cadre de l'extension de classe, il doit être pris en compte dans la conception générale de l'application.

Détermination pour l'extension

XDF tente d'étendre une classe lorsque les noms de classe correspondent et que les noms des zones ne génèrent pas de conflits de zones. Les annotations ClassAlias et FieldAlias sont utiles lorsque vous voulez faire correspondre des classes entre des applications C# et Java dans lesquelles les noms des classes ou les zones diffèrent légèrement. Vous pouvez placer ces annotations dans l'application Java ou C# ou les deux applications. Cependant, la recherche de la classe dans l'application Java peut s'avérer moins efficace que de définir ClassAlias dans l'application C#. Pour plus d'informations sur les annotations ClassAlias et FieldAlias, voir Annotations ClassAlias et FieldAlias

Impact des zones manquantes dans les données sérialisées

Le constructeur de la classe n'est pas appelé au cours de la sérialisation. Par conséquent, les zones manquantes ont une valeur par défaut qui lui est affectée en fonction du langage. L'application qui ajoute de nouvelles zones doit pouvoir détecter les zones manquantes et réagir lorsqu'une ancienne version de la classe est extraite.

Pour que les anciennes applications conservent les nouvelles zones, la seule solution consiste à mettre à jour les données.

Une application peut exécuter une extraction et mettre à jour la mappe avec une ancienne version de la classe qui ne contient pas certaines zones dans la valeur sérialisée depuis le client. Le serveur fusionne les valeurs sur le serveur et détermine si des zones de la version d'origine sont fusionnées dans le nouvel enregistrement. Si une application exécute une extraction, puis supprime et insère une entrée, les zones de la valeur d'origine sont perdues.

Fusion des fonctions

Les options dans une matrice ou une collecte ne sont pas fusionnées par XDF. Il n'est pas toujours aisé de déterminer si une mise à jour dans une matrice ou une collecte doit changer les éléments de lamatrice ou du type. Si une fusion se produit en fonction du positionnement, lorsqu'une entrée dans la matrice est déplacée, XDF peut fusionner les zones qui doivent être associées. Par conséquent, XDF ne tente pas de fusionner le

contenu des matrices ou des collectes. Cependant, si vous ajoutez une matrice dans une nouvelle version d'une définition de classe, la matrice est de nouveau fusionnée dans la version précédente de la classe.

Rubrique parent : Développement d'applications de grille de données d'entreprise

Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Utilisez des annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes Java™ et .NET.

Avant de commencer

- Vous devez avoir configuré IBM® eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).

Pourquoi et quand exécuter cette tâche

Vous pouvez envisager d'utiliser des annotations ClassAlias et FieldAlias si vous disposez d'une classe Java et voulez créer une classe C# correspondante. Dans ce scénario, vous pouvez ajouter des annotations à la classe C#, qui contiennent le nom de classe Java. Pour plus d'informations sur les annotations ClassAlias et FieldAlias, voir Annotations ClassAlias et FieldAlias.

Procédure

Utilisez des annotations ClassAlias et FieldAlias pour corréler des objets entre une classe Java et une classe C#. Java

Figure 1. Exemple Java avec des annotations ClassAlias et FieldAlias

```
@ClassAlias("Employee")
class com.company.department.Employee {

     @FieldAlias("id")
     int myId;

      String name;
}
```

.NET

Figure 2. Exemple .NET avec des attributs ClassAlias et FieldAlias

```
[ ClassAlias( "Employee") ]
class Com.MyCompany.Employee {

     [ FieldAlias("id") ]
      int identifier;

      string name;
}
```

Annotations ClassAlias et FieldAlias
Utilisez les annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes. Vous pouvez partager des données entre deux classes Java, ou entre une classe Java et une classe .NET.

.NET **Annotations ClassAlias et FieldAlias**
Utilisez les annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes. Vous pouvez partager des données entre deux classes Java, ou entre une classe Java et une classe .NET.

Rubrique parent : Développement d'applications de grille de données d'entreprise
Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Concepts associés:
Annotations ClassAlias et FieldAlias

Référence associée:
Fichier de propriétés du client

Information associée:
Leçon 2.3 : Création d'une application de grille de données

Annotations ClassAlias et FieldAlias

Utilisez les annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes. Vous pouvez partager des données entre deux classes Java™, ou entre une classe Java et une classe .NET.

Si vous définissez deux classes avec le même nom et zones, les données de la grille de données sont automatiquement partagées entre les classes. Par exemple, si vous disposez d'une classe Client1 dans votre application Java et d'une classe Client1 dans votre application .NET contenant les mêmes zones, les données sont partagées entre les classes. Cet exemple suppose que le nom de classe contient également le qualificateur de classe, qui est également le nom de package dans Java, et l'espace-noms en C#. Le nom du package et de l'espace-noms sont automatiquement partagés car ces noms correspondent. Voir l'exemple suivant, dans lequel les deux noms sont insensibles à la casse :

```
Java:
package com.mycompany.app
public class SampleClass {
int field1;
String field2;
}

C#
namespace Com.MyCompany.App
public class SampleClass {
int field1;
string field2;
}
```

Cependant, vous pouvez également corréler des données entre des classes ayant des noms différents. Pour corréler les données à stocker dans la grille de données entre des classes portant des noms différents, utilisez des annotations ClassAlias ou FieldAlias.

Entre deux applications Java : vous pouvez définir deux classes avec des noms différents dans des environnements d'application Java distincts. En marquant les classes avec la même annotation ClassAlias, toutes les zones et tous les types de zone sont en correspondance entre ces deux classes. Les classes sont corrélées avec le même ID de type de classe même si elles portent des noms de classe différents. Le même ID de type de classe et les métadonnées peuvent être réutilisés ensuite dans des environnements d'exécution d'application Java différents.

Entre une application Java et une application .NET : vous pouvez utiliser des annotations similaires dans une application C# pour corréler la classe C# avec une classe Java. Les attributs ClassAlias définis pour la classe C# et les zones sont mis en correspondance avec une classe Java ayant la même annotation ClassAlias.

Rubrique parent : Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET
Rubrique parent : .NET Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Tâches associées:
Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Référence associée:
Fichier de propriétés du client

Information associée:
Leçon 2.3 : Création d'une application de grille de données

Mappage de clés avec des partitions avec des annotations PartitionKey

Un alias PartitionKey est utilisé pour identifier les zones ou les attributs sur lesquels un calcul de code de hachage est exécuté pour déterminer la partition dans laquelle les données sont sauvegardées. L'annotation PartitionKey n'est valide que sur les attributs de clé.

Avant de commencer

Vous devez utiliser eXtreme Data Format (XDF). XDF est activé lorsque vous utilisez IBM eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).

Pourquoi et quand exécuter cette tâche

Vous définissez un alias PartitionKey pour que plusieurs classes enregistrent les données dans la même partition. Par exemple, si vous définissez la valeur PartitionKey comme clé departmentID, les enregistrements d'employé sont placés dans la même partition.

L'interface PartionableKey est l'interface Java existante et elle est prioritaire sur l'annotation PartitionableKey dans C#.

Procédure

- Java Définissez des annotations PartitionKey dans une zone dans une application Java.

  ```
  class Employee {
      int empId;

      @PartitionKey(order = 0)
      int deptId;

         }
  ```

  Vous pouvez définir des annotations PartitionKey sur plusieurs clés ou l'alias PartitionKey dans une classe. Pour d'autres exemples sur la définition des annotations PartitionKey dans les applications Java, voir Documentation des API Java : type d'annotation PartitionKeys.

- .NET Définissez des attributs PartitionKey dans une zone dans une application .NET.

  ```
  class Employee {
      int empId;

      [PartitionKey]
      int deptId;
  }
  ```

  Vous pouvez également définir des attributs PartitionKey dans des classes .NET. Pour plus d'informations, voir Documentation des API .NET : classe PartitionKeyAttribute.

Rubrique parent : Développement d'applications de grille de données d'entreprise
Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Types de données équivalents entre Java et C#

Lorsque vous développez des applications de grille de données d'entreprise, les types de données entre les applications Java et C# doivent être compatibles.

Tableau 1. Types de données équivalents entre Java et C#

Type Java	Type C#
boolean	bool
java.lang.Boolean	bool?
byte	sbyte or byte
java.lang.Byte	sbyte?
short	short?, ushort
java.lang.Short	short?, ushort?
int	int, uint, ushort
java.lang.Integer	int?, uint?
long	long, ulong, uint
java.lang.Long	long?, ulong?, uint?
short ou int	ushort
java.lang.Short ou java.lang.Integer	ushort?
int ou long	uint
java.lang.Integer ou java.lang.Long	uint?
long ou BigInteger	ulong
java.lang.Long ou java.lang.BigInteger	ulong?
char, java.lang.Character	char
java.lang.Character	char?
float, java.lang.Float	float
java.lang.Foat	float?
double	double
java.lang.Double	double?
java.math.BigDecimal	decimal or decimal?
java.math.BigInteger	decimal, long or ulong?
java.lang.String	string
java.util.Date, java.util.Calendar	System.DateTime
java.util.Date(rounding), java.util.Calendar(rounding)	System.DateTime
java.util.ArrayList	System.Collections.ArrayList, System.Collections.Generic.List
java.util.HashMap	System.Collections.Generic.Dictionary, System.Collections.Hashtable
java.util.LinkedList	System.Collections.Generic.LinkedList
java.util.ArrayList, java.util.Vector	System.Collections.Generic.List
java.util.Stack	System.Collections.Generic.Stack
java.util.Vector	System.Collections.ArrayList, System.Collections.Generic.List

Rubrique parent : Développement d'applications de grille de données d'entreprise

Sécurisation de WebSphere DataPower XC10 Appliance

Les grilles de données du dispositif stockent des informations qui sont confidentielles et doivent être protégées.

Avant de commencer

Certains aspects de ce scénario, tels que l'activation de la norme FIPS (Federal Information Processing Standard) 140-2, nécessitent que tous les membres d'une collectivité soient au niveau le plus récent. Si le dispositif à sécuriser fait partie d'une collectivité, le microprogramme de tous les membres de cette collectivité doit être mis à niveau pour que vous puissiez effectuer les tâches de ce scénario.

Pourquoi et quand exécuter cette tâche

WebSphere DataPower XC10 Appliance comprend des contrôles de sécurité globaux. La configuration par défaut comporte des mots de passe par défaut, des clés SSL et des valeurs d'authentification confidentielles que vous devez modifier. Suivez ce scénario pour modifier la configuration, puis procédez au déploiement du dispositif sécurisé.

Pour un déploiement sécurisé, utilisez plusieurs couches de protection pour optimiser la sécurité. Le premier élément de protection consiste à utiliser des pare-feu pour segmenter le réseau. Le modèle standard à niveaux pour les applications Web est constitué des clients, d'un niveau présentation constitué de serveurs HTTP, d'un niveau application constitué des serveurs d'application, d'un niveau données et d'un niveau stockage.

Les dispositifs WebSphere DataPower XC10 Appliance sont déployés dans le niveau données. En règle générale, il convient de placer les serveurs de la couche présentation dans une zone démilitarisée (DMZ) protégée par un pare-feu, et de placer les niveaux application, données et stockage dans des segments du réseau protégés par d'autres pare-feu. Ne déployez pas un dispositif dans une zone démilitarisée. Vous devez protéger vos dispositifs comme tous les autres éléments du niveau données, conformément aux bonnes pratiques communément utilisées.

Cependant, pour optimiser la protection contre les menaces à la sécurité, utilisez un mécanisme de défense efficace, comprenant des mesures supplémentaires qui protègent le fonctionnement des dispositifs et les données stockées dans la grille de données. Ces mesures supplémentaires offrent une protection contre les attaques externes, mais interdisent également les accès non autorisés aux données par les employés et les sous-traitants qui pourraient avoir accès aux segments du réseau contenant les dispositifs.

Les différentes étapes de ce scénario sont effectuées dans la console Web de WebSphere DataPower XC10 Appliance. Elles peuvent également être automatisées en appelant l'interface de ligne de commande HTTP à partir d'un programme. Pour plus d'informations concernant l'interface de ligne de commande HTTP, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

1. 2.5+ Configuration d'un accès sécurisé à la grille de données
   Configurez un accès administrateur et spécifiez des paramètres pour vos collectivités de dispositifs afin de configurer une authentification et une autorisation sécurisées pour l'accès à la grille de données.

2. 2.5+ Activation de la sécurité pour les grilles de données
   Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

3. 2.5+ Configuration de la sécurité du client
   Une fois que vous avez sécurisé la grille de données sur le dispositif, vous devez configurer les clients afin qu'ils puissent se connecter à la grille de données sécurisée.

Rubrique parent : 2.5+ Scénarios

Configuration d'un accès sécurisé à la grille de données

Configurez un accès administrateur et spécifiez des paramètres pour vos collectivités de dispositifs afin de configurer une authentification et une autorisation sécurisées pour l'accès à la grille de données.

Pourquoi et quand exécuter cette tâche

Pour une protection optimale contre les menaces pour la sécurité, suivez la procédure ci-dessous afin de réduire les menaces externes et les accès non autorisés aux données par les employés et les sous-traitants qui pourraient avoir accès aux segments du réseau dans lesquels les dispositifs communiquent avec les clients et les serveurs.

Avertissement : Dans la console d'administration, lorsque vous spécifiez les paramètres de collectivité pour la communication de serveur à serveur et la protection des données sur le réseau, vous pouvez configurer les paramètres qui contrôlent la sécurité. Si vous modifiez ces paramètres, vous devez redémarrer l'ensemble de la collectivité. Pour pouvoir définir ces paramètres, cliquez sur **Collectivité** > **Paramètres**. Pour plus d'efficacité, effectuez toutes les modifications en une seule opération afin de ne devoir redémarrer la collectivité qu'une seule fois.

Procédure

1. Sécurisez l'accès administrateur

   Pour pouvoir configurer le dispositif, utilisez l'ID administrateur `xcadmin` et le mot de passe d'administration par défaut `xcadmin`. Cet ID utilisateur et ce mot de passe donnent un accès complet à l'ensemble des fonctions d'administration et des données du dispositif et de la collectivité. Il est donc très important de configurer un mot de passe d'administration racine qui soit difficile à deviner. Pour ce faire, dans la console Web, cliquez sur **Collectivité** > **Utilisateurs**. Sélectionnez l'utilisateur **Administrateur** et modifiez son mot de passe.

2. Configurez la sécurité du transport SSL

   Le dispositif est configuré avec un fichier de clés et un fichier de clés certifiées par défaut. Le fichier de clés certifiées par défaut inclut le certificat de signataire du fichier de clés par défaut. Etant donné que ce certificat est livré avec chaque dispositif, il doit être remplacé pour que la configuration SSL soit sécurisée. Le remplacement du certificat se déroule en plusieurs étapes.

   a. Créez un nouveau certificat et une clé privée Ce certificat peut être autosigné, mais il est préférable qu'il soit émis par une autorité de certification locale.
   b. Créez ou modifiez le fichier de clés certifiées Une méthode consiste à télécharger le fichier de clés certifiées par défaut à partir du dispositif afin de supprimer le certificat de signataire du fichier de clés certifiées libellé **IBM WebSphere DataPower XC10**, puis à ajouter un certificat afin d'établir une relation de confiance avec le nouveau fichier de clés. Si le nouveau fichier de clés comporte un certificat autosigné, ce certificat doit être importé dans le fichier de clés certifiées. Si le nouveau fichier de clés comporte un certificat émis par une autorité de certification locale, le certificat racine de cette autorité doit être ajouté au fichier de clés certifiées.
   c. Téléchargez les nouveaux fichiers de clés et de clés certifiées.
   d. Modifiez le paramètre d'alias de certificat afin qu'il corresponde au nom du certificat dans le fichier de clés. Les fichiers de clés certifiées de tous les clients de grille de données doivent également être mis à jour pour établir une relation de confiance avec le certificat du dispositif.

      Gérez le fichier de clés et le fichier de clés certifiées à l'aide de la commande keytool fournie avec l'environnement d'exécution Java (JRE, Java Runtime Environment). Vous pouvez aussi utiliser d'autres outils de gestion de clés. Reportez-vous aux tâches administratives ci-après, qui permettent de gérer les fichier de clés et les fichiers de clés certifiées.

      - Générez un certificat autosigné.

        ```
        keytool -genkey -alias ogsample -keystore key.jks -storetype JKS -keyalg
        rsa -dname "CN=ogsample, OU=OGSample, O=acme, L=Your City, S=Your State,
        C=Your Country" -storepass ogpass
        -keypass ogpass -validity 3650
        ```

      - Supprimez le certificat de signataire par défaut du dispositif du fichier de clés certifiées téléchargé à partir du dispositif.

        ```
        keytool -delete -alias “ibm websphere datapower xc10” -keystore trust.jks
        ```

```
-storetype JKS -keypass xc10pass
```

- Exportez le nouveau certificat du fichier de clés.

  ```
  keytool -export -alias ogsample -keystore key.jks -file temp.key -
  storepass ogpass
  ```

- Importez le certificat dans le fichier de clés certifiées.

  ```
  keytool -import -noprompt -alias ogsamplepublic -keystore trust.jks -file
  temp.key -storepass xc10pass
  ```

  Cette étape d'importation est nécessaire pour configurer le fichier de clés certifiées afin qu'il puisse établir une relation de confiance avec un certificat exporté à partir d'un fichier de clés. Cette étape est effectuée dans plusieurs contextes. Le fichier de clés certifiées du client doit être mis à jour pour qu'une relation de confiance puisse être établie avec le certificat du fichier de clés du dispositif, et, dans certaines configurations, le fichier de clés certifiées du dispositif doit être mis à jour pour qu'une relation de confiance puisse être établie avec le certificat du fichier de clés du client. L'idéal serait que les certificats du fichier de clés du client et du dispositif soient émis par une autorité de certification locale, auquel cas la confiance sera établie par l'importation du certificat de signataire racine de cette autorité de certification.
  - Modifiez le mot de passe dans le fichier de clés et le fichier de clés certifiées.

    ```
    keytool -storepasswd -new newpass -keystore trust.jks -storepass xc10pass
    ```

  - Téléchargez les nouveaux fichiers de clés et de clés certifiées vers la collectivité de dispositifs.

e. Spécifiez un mode SSL parmi les options suivantes : **TCP/IP**, où SSL n'est pas utilisé pour la grille de données de communication, **TLS pris en charge** et **TLS requis**.

   TLS requis est recommandé pour la sécurité. Si TLS n'est pas utilisé, les mots de passe et les données de grille sensibles sont transmises non chiffrées sur les liaisons réseau qui relient les clients de la grille et le dispositif.

   Remarque : Lorsque la communication TLS est obligatoire, le fichier de clés certifiées du dispositif doit être configuré pour l'établissement d'une relation de confiance avec le certificat contenu dans le fichier de clés du client, et le fichier de clés certifiées du client doit être configuré pour l'établissement d'une relation de confiance avec le certificat contenu dans le fichier de clés du serveur. Voir Configuration de la sécurité du client.

f. Activez la norme FIPS (Federal Information Processing Standard).

   Vous pouvez configurer la collectivité de dispositifs pour qu'elle utilise la norme FIPS 140-2 pour toutes les communications réseau chiffrées. Cette norme garantit une protection élevée des données transmises. Sélectionnez **l'option d'activation de la cryptographie FIPS 140-2** pour activer FIPS.

   Certaines versions de navigateur Web ne fonctionnent pas avec un serveur sur lequel FIPS est activée. Toutefois, les versions actuelles de la plupart des navigateurs (y compris Mozilla Firefox, Microsoft Internet Explorer et Google Chrome) prennent en charge la communication avec ce type de serveur. Vous devrez peut-être configurer votre navigateur de manière à activer TLS, car sslv3 n'est pas pris en charge en mode FIPS. Pour plus d'informations, voir la documentation de votre navigateur.

g. Activez l'authentification par certificat client.

   Lorsque cette option est activée, le fichier de clés de chaque client et navigateur qui communique avec le dispositif doit être configuré pour comporter un certificat accepté par le fichier de clés certifiées du dispositif. L'authentification par certificat client n'est pas utilisée pour le transport ORB. Elle n'est utilisée que pour le transport HTTPS et XIO. Il n'est pas nécessaire d'activer l'authentification par certificat client pour avoir une configuration sécurisée.

3. Configurez l'authentification de serveur à serveur.

   La communication de grille de données entre les dispositif d'une collectivité, et entre des domaines de dispositif liés, est authentifiée à l'aide d'une clé secrète partagée. Les dispositifs sont déjà configurés avec une clé secrète par défaut codée en dur. Vous devez modifier cette clé secrète afin de disposer d'une configuration sécurisée. Pour pouvoir spécifier une clé secrète unique, sélectionnez **Remplacer la clé secrète d'authentification définie par défaut en usine (recommandé)**.

Cette clé secrète doit être une phrase de passe longue et difficile à deviner. Notez-la et stockez-la dans un endroit sûr. Lorsque les collectivités sont jointes dans des domaines liés, chaque collectivité doit être configurée avec la même clé secrète.

4. Exigez l'authentification pour toutes les requêtes concernant la grille de données.

   Vous pouvez configurer l'authentification pour chaque requête client à la grille de données. Par défaut, cette authentification n'est pas obligatoire. Toutefois, vous pouvez protéger l'accès aux grilles de données en sécurisant chaque grille individuellement, et sécuriser la configuration en exigeant l'authentification pour toutes les requêtes aux grilles. Lorsque cette option est définie, chaque client doit être configuré avec un ID utilisateur et un mot de passe connus de la collectivité de dispositifs, ou, dans le cas de l'authentification LDAP, enregistrés dans LDAP. Seul l'administrateur racine (ID utilisateur xcadmin) peut se connecter sans passer par l'authentification LDAP.

   Une fois effectuées les étapes 2 à 4, soumettez les modifications pour redémarrer la collectivité. Les nouvelles valeurs sont automatiquement propagées aux dispositifs qui doivent être assimilés dans la collectivité. Si vous avez activé la norme FIPS, elle est activée sur chaque dispositif avant son assimilation dans la collectivité.

5. Désactivez l'accès Telnet.

   Le configuration par défaut du dispositif comprend un serveur Telnet actif. La communication Telnet du dispositif ne prend pas en charge SSL. Pour sécuriser la configuration, désactivez le serveur Telnet. Pour ce faire, établissez une session SSH avec le dispositif en utilisant l'ID utilisateur et le mot de passe de l'administrateur, puis lancez la commande **`platform service telnet disable`**. Cette commande n'agissant pas sur l'ensemble de la collectivité, vous devez l'exécuter sur chaque dispositif. La commande **`platform service telnet enable`** démarre Telnet s'il a été désactivé. Cette procédure est manuelle et ne peut être automatisée.

6. Configurez l'authentification LDAP.

   L'authentification pour le navigateur, pour REST et pour l'accès de la grille au dispositif peut s'effectuer de deux manières : vous pouvez enregistrer les identités authentifiées dans la collectivité de dispositifs ou dans LDAP. Vous pouvez utiliser l'une ou l'autre de ces deux méthodes dans une configuration sécurisée.

   Conseil : Notez que l'authentification de l'administrateur racine utilise toujours un mot de passe vérifié par la collectivité et non par LDAP.

   Lorsque l'authentification LDAP est utilisée, protégez la connexion LDAP par SSL afin que les mots de passe ne circulent pas en clair sur le réseau. Pour activer SSL pour une connexion LDAP, spécifiez une URL LDAPS, telle que `ldaps://ldapserver.company.com:636`. Maintenant, vous devez configurer le fichier de clés certifiées du dispositif avec un certificat afin d'établir une relation de confiance avec le serveur LDAP en vue de la communication SSL. Pour plus d'informations sur la configuration de l'authentification LDAP, voir Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP.

Rubrique parent : 2.5+ Sécurisation de WebSphere DataPower XC10 Appliance
Rubrique suivante : 2.5+ Activation de la sécurité pour les grilles de données

Information associée:
Configuration des fichiers JSSE conformes à la norme FIPS

Activation de la sécurité pour les grilles de données

Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

Pourquoi et quand exécuter cette tâche

Lorsqu'une grille de données appartient à une collectivité, vous pouvez configurer la sécurité et le contrôle d'accès pour cette grille en particulier. L'activation de la sécurité permet d'imposer aux clients qui tentent d'accéder à la grille de s'authentifier. Indépendamment de ce paramètre, si la collectivité est configurée pour exiger l'authentification pour tous les accès aux grilles de données, l'authentification est toujours exigée pour l'accès à cette grille spécifique. Mais exiger l'authentification peut ne pas restreindre suffisamment les accès aux grilles. Par exemple, dans le cas de l'authentification LDAP, n'importe quel utilisateur LDAP peut accéder à cette grille spécifique. Pour en restreindre l'accès, vous pouvez activer l'autorisation pour cette grille.

Important : Si vous modifiez les paramètres de sécurité d'une grille de données, cette dernière redémarre automatiquement. Lorsque la grille de données redémarre, les données qui s'y trouvent sont perdues. Par conséquent, configurez la sécurité de vos grilles de données avant de commencer à y sauvegarder des données.

La communication via la passerelle REST est toujours sécurisée, même si la sécurité n'est pas activée sur la grille de données. Pour plus d'informations, voir Passerelle REST : Configuration de la sécurité.

Procédure

1. Dans l'interface utilisateur, accédez aux paramètres de la grille de données. Cliquez sur **Grille de données** > ***type_grille_données***. Cliquez sur le nom de la ***nom_grille_données*** à modifier.
2. Activez la sécurité ou l'autorisation pour la grille de données. Cliquez sur **Activer la sécurité** pour permettre à tous les utilisateurs pouvant accéder à l'interface utilisateur d'accéder à la grille de données. Si vous souhaiter restreindre davantage l'accès, cliquez sur **Activer l'autorisation**.

   Remarque : Vous devez activer l'autorisation pour une grille de données de session.

   Une fois l'autorisation activée, vous pouvez spécifier une liste d'utilisateurs ou de groupe d'utilisateurs dans la liste**Accès accordé à**. Lorsque l'autorisation est activée, seuls les utilisateurs figurant dans cette liste sont autorisés à accéder aux données de la grille de données. Vous pouvez affecter les accès suivants aux utilisateurs ou aux groupes d'utilisateurs en cliquant sur le nom du type d'accès par défaut qui est affiché dans l'interface utilisateur :
   - **read** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire ou interroger des données de la grille de données.
   - **write** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire, interroger et écrire des données dans la grille de données.
   - **create** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire, interroger, écrire, insérer et créer des mappes dynamiques dans la grille de données.
   - **all** : lorsque ce droit est affecté, l'utilisateur ou le groupe d'utilisateurs peut lire, interroger, écrire, insérer, créer des mappes dynamiques, supprimer et invalider des données de la grille de données. Les administrateurs de dispositif disposent du droit **all** par défaut.

   Lorsque vous changez les paramètres de sécurité et d'autorisation, le délai d'attente est de cinq minutes.
   - **Délai d'authentification** : si vous changez le mot de passe d'un utilisateur déjà authentifié dans la grille de données, les données d'identification d'origine sont toujours valides pendant 5 minutes.
   - **Délai d'autorisation** : si vous supprimez une autorisation pour un utilisateur, ce dernier reste autorisé pendant cinq minutes. Ce délai ne s'applique qu'aux autorisations supprimées. Si vous ajoutez une autorisation à un utilisateur, l'utilisateur est autorisé immédiatement.

Rubrique parent : 2.5+ Sécurisation de WebSphere DataPower XC10 Appliance
Rubrique précédente : 2.5+ Configuration d'un accès sécurisé à la grille de données
Rubrique suivante : 2.5+ Configuration de la sécurité du client
Rubrique parent : Configuration des grilles de données
Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Configuration de la sécurité du client

Une fois que vous avez sécurisé la grille de données sur le dispositif, vous devez configurer les clients afin qu'ils puissent se connecter à la grille de données sécurisée.

Pourquoi et quand exécuter cette tâche

Si le dispositif est configuré pour exiger TLS, le client doit être configuré pour le transport SSL et il doit avoir un fichier de clés et un fichier de clés certifiées appropriés. Si une authentification est requise, le client doit également être configuré avec un ID utilisateur et un mot de passe. La procédure à suivre varie selon que l'installation du client s'exécute de façon autonome ou dans un processus WebSphere Application Server.

Dans le cas d'un environnement autonome, vous devez configurer un fichier `client.properties` contenant les paramètres à transmettre à l'application de grille de données. Si votre environnement inclut WebSphere Application Server, utilisez les outils de WebSphere Application Server pour configurer la sécurité du client.

Procédure

1. Si le client est une installation autonome, utilisez un fichier de propriétés du client pour configurer la sécurité du client et la communication entre le client et le serveur. Pour plus de détails concernant l'emplacement et le format de ce fichier, voir IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité.

   Pour connaître les propriétés de sécurité que vous pouvez configurer pour les clients Java™ ou .NET, voir Fichier de propriétés du client.

2. Si le client s'exécute dans un processus WebSphere Application Server, vous devez configurer les valeurs de configuration SSL (y compris le fichier de clés et le fichier de clés certifiées) à l'aide des outils de WebSphere Application Server.

   Un utilitaire fourni permet d'importer le certificat à partir du fichier de clés du dispositif dans le fichier de clés certifiées de WebSphere Application Server. Pour plus de détails, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

   Vous devez configurer ce fichier de clés certifiées de WebSphere Application Server pour qu'il accepte le certificat du fichier de clés du dispositif. Si le dispositif est configuré pour l'authentification par certificat client, ou si la communication via ORB (Object Request Broker) est utilisée, le fichier de clés certifiées du dispositif doit aussi être configuré pour accepter le certificat du fichier de clés de WebSphere Application Server.

   Si le dispositif est configuré pour exiger l'authentification, le client du dispositif qui s'exécute sous WebSphere Application Server doit aussi être configuré pour nécessiter une authentification à l'aide de données d'identification. Vous pouvez configurer le client avec un fichier de propriétés, comme pour les clients autonomes Java. Toutefois, lorsque le client du dispositif est installé sur WebSphere Application Server, la console d'administration de WebSphere Application Server est modifiée pour vous permettre d'entrer les données d'identification. Vous pouvez ainsi configurer les domaines de service de catalogue et spécifier les propriétés d'authentification du client.

3. Si le client s'exécute dans un environnement dans lequel la norme FIPS (Federal Information Processing Standard) est activée sur le serveur, il doit utiliser le protocole d'établissement de liaison TLS pour communiquer avec le dispositif. Il n'est pas possible dans ce cas d'utiliser le protocole d'établissement de liaison SSL. Le protocole TLS se configure en indiquant `protocol=TLS` dans le fichier de propriétés du client.

   - Lorsque XIO est utilisé et que le fichier de propriétés du client ne contient pas de paramètre de protocole, la valeur par défaut est SSL et TLS. Ce paramétrage fonctionne donc lorsque la norme FIPS est activée sur le dispositif.
   - Lorsque la communication de type ORB est utilisée sur un client autonome, le protocole doit être défini sur TLS dans le fichier de propriétés du client pour permettre la communication avec un dispositif en mode FIPS.
   - Lorsque la communication de type ORB est utilisée et que le client s'exécute avec WebSphere Application Server, ce sont les paramètres de sécurité de WebSphere qui sont utilisés pour définir le protocole d'établissement de liaison. Par défaut, il prend la valeur SSL_TLS, qui est appropriée lorsque la norme FIPS est activée sur le dispositif.

     Ce paramètre peut être configuré sur TLS ou SSL_TLS à l'aide de la console d'administration de WebSphere Application Server. Cliquez sur **Sécurité** > **Certificat SSL et gestion des clés** > **Gérer les configurations de sécurité des noeuds finals**. Sélectionnez un nom de cellule et cliquez sur **Configurations SSL** > **CellDefaultSSLSettings** > **Paramètres de la qualité de protection (QoP)**. Assurez-vous que la zone de protocole contient bien TLS ou SSL_TLS.

Rubrique parent : 2.5+ Sécurisation de WebSphere DataPower XC10 Appliance
Rubrique précédente : 2.5+ Activation de la sécurité pour les grilles de données

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Référence associée:
Fichier de propriétés du client

Java

Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Vous pouvez migrer une session de réplication de mémoire à mémoire ou une session de base de données pour utiliser la gestion de session WebSphere eXtreme Scale.

Avant de commencer

- Pour le support de session des applications client exécutées sur WebSphere Application Server dans le cluster, WebSphere eXtreme Scale doit être installé sur les déploiements de noeud WebSphere Application Server, y compris le noeud de gestionnaire de déploiement. Voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Les étapes dans ce scénario s'appliquent à la version 8.5 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction la version WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Dans le cadre de la migration vers une session WebSphere DataPower XC10 Appliance, vous devez noter les paramètres de configuration précédents dans la console d'administration WebSphere Application Server. Lors de la migration vers une session WebSphere DataPower XC10 Appliance, les paramètres de configuration doivent refléter ce que vous avez déjà configuré pour la base de données ou la session mémoire à mémoire.

Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Dans le cadre d'une migration vers une session HTTP dans la grille de données, vous devez créer un domaine de service de catalogue à l'aide de la console d'administration de WebSphere Application Server.

Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Vous devez utiliser les paramètres de configuration précédents que vous avez notés dans la console d'administration WebSphere Application Server pour associer une application ou un serveur d'applications à la gestion de session WebSphere eXtreme Scale.

Rubrique parent : 2.5+ Scénarios
Rubrique parent : Création de grilles de données de session

Tâches associées:
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

Java

Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server

Dans le cadre de la migration vers une session WebSphere DataPower XC10 Appliance, vous devez noter les paramètres de configuration précédents dans la console d'administration WebSphere Application Server. Lors de la migration vers une session WebSphere DataPower XC10 Appliance, les paramètres de configuration doivent refléter ce que vous avez déjà configuré pour la base de données ou la session mémoire à mémoire.

Pourquoi et quand exécuter cette tâche

La console d'administration WebSphere Application Server contient des paramètres spécifiques que vous devez noter. Vous en aurez besoin lors de la mise à jour du fichier `splicer.properties`. Les étapes de cette procédure s'appliquent à la version 8.5 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction de la version WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Client version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Procédure

1. Démarrez la console d'administration WebSphere Application Server.
    - Si vous avez déjà défini des paramètres au niveau du serveur, accédez à :
        a. **Serveurs > Types de serveur > Serveurs d'applications WebSphere**
        b. Dans la zone **Serveurs d'applications**, sélectionnez **le nom du serveur**
        c. Dans la zone **Paramètres de conteneur**, cliquez sur **Gestion de session**/
    - Si vous avez déjà défini des paramètres au niveau de l'application, accédez à :
        a. **Applications > Toutes les applications**.
        b. Dans la zone **Serveurs d'applications**, sélectionnez le **nom de l'application**.
        c. Dans la zone **Propriétés du module Web**, cliquez sur **Gestion des sessions**.
2. Dans les **propriétés générales**, cochez la case **d'autorisation de dépassement**.
3. Dans la zone des **propriétés générales**, notez les paramètres WebSphere Application Server. Vous en aurez besoin lors de la mise à jour les propriétés dans le fichier `splicer.properties`.

    Tableau 1. Paramètres de configuration pour mettre à jour le fichier splicer.properties

Paramètres dans la console d'administration WebSphere Application Server	Propriétés à mettre à jour dans le fichier splicer.properties
Activer les cookies	useCookies
Activer la réécriture des URL	useURLEncoding
Nombre maximal de sessions en mémoire	sessionTableSize

4. Dans la zone des **propriétés générales**, si la case **Activer les cookies** est sélectionnée, cliquez dessus et notez les paramètres WebSphere Application Server. Vous en aurez besoin pour mettre à jour les propriétés dans le fichier `splicer.properties`.

    Tableau 2. Paramètres de configuration dans le fichier splicer.properties

Paramètres dans la console d'administration WebSphere Application Server	Propriétés à mettre à jour dans le fichier splicer.properties
Domaine du cookie	cookieDomain
Chemin du cookie	cookiePath

5. Cliquez sur **Gestion des sessions**, puis dans la zone des **propriétés supplémentaires**, cliquez sur **Distributed environment settings**.
6. Dans la zone **Distributed Sessions** remplacez la base de données ou la configuration de réplication de mémoire à mémoire par **None**.
7. Cliquez sur **Custom Tuning Properties** et notez les paramètres WebSphere Application Server. Vous en aurez besoin pour mettre à jour les propriétés dans le fichier `splicer.properties`.

    Tableau 3. Paramètres de configuration des propriétés dans le fichier splicer.properties

Paramètres dans la console d'administration WebSphere Application Server	Propriétés à mettre à jour dans le fichier splicer.properties
Fréquence d'écriture	replicationInterval

Fréquence d'écriture	replicationInterval
Write contents	fragmentedSession

Que faire ensuite

Ensuite, créez le domaine de service de catalogue pour une session WebSphere DataPower XC10 Appliance.

Rubrique parent : Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

Java

Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données

Dans le cadre d'une migration vers une session HTTP dans la grille de données, vous devez créer un domaine de service de catalogue à l'aide de la console d'administration de WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Cette procédure s'applique à la version 8.5 de la console d'administration de WebSphere Application Server. Ces informations peuvent varier en fonction de la version de WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Client version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Créez le domaine de service de catalogue pour la grille de données à l'aide de la console d'administration de WebSphere Application Server. Pour plus d'informations, voir Création de domaines de service de catalogue dans WebSphere Application Server.

Procédure

1. Démarrez la console d'administration WebSphere Application Server.
2. Dans le menu supérieur, cliquez sur **Administration de système > WebSphere eXtreme Scale > Domaines de service de catalogue**

   Remarque : Si WebSphere eXtreme Scale n'apparaît pas, cela signifie que votre profil WebSphere Application Server n'a pas été étendu pour la grille de données. .
3. Cliquez sur **Nouveau**.
4. Définissez le nom du service de catalogue dans la zone **Nom**.
5. Dans la zone **Serveurs de catalogue**, choisissez **Serveur distant** et définissez l'emplacement ou le nom du serveur distant dans la zone.
6. Définissez le numéro de port dans la zone **Port d'écoute**.
7. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.

Que faire ensuite

Ensuite, utilisez les paramètres de configuration précédents que vous avez notés dans la console d'administration WebSphere Application Server pour associer une application ou un serveur d'applications à la gestion de session WebSphere eXtreme Scale.

Rubrique parent : Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

Java

Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents

Vous devez utiliser les paramètres de configuration précédents que vous avez notés dans la console d'administration WebSphere Application Server pour associer une application ou un serveur d'applications à la gestion de session WebSphere eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Cette procédure s'applique à la version 8.5 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction la version WebSphere Application Server que vous utilisez.

Remarque : WebSphere eXtreme Scale Client Version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Procédure

- Si vous voulez configurer une application pour l'associer à la gestion de session WebSphere eXtreme Scale, procédez comme suit :
    1. Démarrez la console d'administration WebSphere Application Server.
    2. Dans le menu supérieur, cliquez sur **Applications > Toutes les applications**.
    3. Dans la zone **Applications d'entreprise WebSphere**, sélectionnez le **nom d'application**.
    4. Dans la zone des propriétés **Module Web**, cliquez sur **Gestion de session**.
    5. Cliquez sur **Paramètres de gestion de session eXtreme Scale**.
    6. Si WebSphere eXtreme Scale n'apparaît pas, cela implique que votre profil WebSphere Application Server n'a pas été étendu pour WebSphere eXtreme Scale. Pour plus d'informations, voir .
    7. Pour configurer une application pour WebSphere eXtreme Scale Client, procédez comme suit :
        a. Dans la liste **>Gérer la persistance des sessions par**, sélectionnez **IBM WebSphere DataPower XC10 Appliance**
        b. Définissez l'adresse IP ou le nom d'hôte du domaine du service de catalogue que vous avez créé depuis la liste.
        c. Définissez un nom d'utilisateur et un mot de passe et testez la connexion.
        d. Définissez une nouvelle grille de données ou sélectionnez une grille de données existante.
    8. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
    9. Un fichier splicer.properties est créé pour l'application. L'emplacement du fichier `splicer.properties` est la valeur d'une nouvelle propriété `{application name},com.ibm.websphere.xs.sessionFilterProps`. Pour rechercher la propriété personnalisée, accédez à **Administration de système> Cellule** et cliquez sur **Propriétés personnalisées**.
    10. Mettez à jour le fichier `splicer.properties` avec les valeurs que vous avez obtenues dans Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server.
    11. Redémarrez les processus serveur d'applications.

    Remarque : Changez `splicer.properties` au niveau du gestionnaire de déploiement pour que les propriétés soient synchronisées sur l'agent de noeud. Si vous mettez à jour `splicer.properties` au niveau du noeud, le gestionnaire de déploiement remplace le fichier `splicer.properties` lors de la synchronisation suivante.

    Remarque : Si vous revenez dans la gestion de session de base de données et la gestion de session WebSphere eXtreme Scale, le fichier `splicer.properties` est recréé et les modifications que vous avez apportées sont remplacées. Pour plus d'informations sur le processus de synchronisation de fichier depuis le gestionnaire de déploiement vers les notes et connaître les éléments modifiés, voir Synchronisation des fichiers de gestion de système.

- Si vous voulez configurer un serveur d'applications pour l'associer à la gestion de session WebSphere eXtreme Scale, procédez comme suit :
    1. Démarrez la console d'administration WebSphere Application Server.
    2. Dans le menu supérieur, cliquez sur **Serveurs > Types de serveur > Serveurs d'applications WebSphere**.
    3. Dans la zone **Serveurs d'applications**, sélectionnez le **nom du serveur**.
    4. Dans la zone **Paramètres de conteneur**, cliquez sur **Gestion de session**/
    5. Cliquez sur **Paramètres de gestion des sessions eXtreme Scale**.

        Remarque : Si WebSphere eXtreme Scale ne s'affiche pas, cela implique que votre profil WebSphere Application Server n'a pas été étendu pour WebSphere eXtreme Scale. Pour plus d'informations, voir .

6. Pour configurer une application pour WebSphere eXtreme Scale Client, procédez comme suit :
    a. Dans la liste **>Gérer la persistance des sessions par**, sélectionnez **IBM WebSphere DataPower XC10 Appliance**
    b. Définissez l'adresse IP ou le nom d'hôte du domaine du service de catalogue que vous avez créé depuis la liste.
    c. Définissez un nom d'utilisateur et un mot de passe et testez la connexion.
    d. Définissez une nouvelle grille de données ou sélectionnez une grille de données existante.
7. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
8. Un fichier splicer.properties est créé pour l'application. L'emplacement du fichier splicer.properties est la valeur d'une nouvelle propriété `com.ibm.websphere.xs.sessionFilterProps`. Pour rechercher la propriété personnalisée, accédez à **Serveurs > Types de serveur > Serveurs d'applications WebSphere**.
9. Dans la zone des **serveurs d'applications**, sélectionnez le **nom du serveur**.
10. Dans la zone **Infrastructure du serveur**, sélectionnez **Propriétés personnalisées**.
11. Mettez à jour le fichier `splicer.properties` avec les valeurs que vous avez obtenues dans Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server.
12. Redémarrez les processus serveur d'applications.

Remarque : Changez `splicer.properties` au niveau du gestionnaire de déploiement pour que les propriétés soient synchronisées sur l'agent de noeud. Si vous mettez à jour `splicer.properties` au niveau du noeud, le gestionnaire de déploiement remplace le fichier `splicer.properties` lors de la synchronisation suivante.

Remarque : Si vous revenez dans la gestion de session de base de données et la gestion de session WebSphere eXtreme Scale, le fichier `splicer.properties` est recréé et les modifications que vous avez apportées sont remplacées. Pour plus d'informations sur le processus de synchronisation de fichier depuis le gestionnaire de déploiement vers les notes et connaître les éléments modifiés, voir Synchronisation des fichiers de gestion de système.

Résultats

Vous venez de changer les paramètres de configuration précédents de la gestion de session de mémoire à mémoire ou de base de données avec la gestion de session WebSphere eXtreme Scale.

Rubrique parent : Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Fichier splicer.properties

| Suivant >

Tutoriel : Démarrer avec des applications de grille de données simples

Vous pouvez utiliser le modèle d'application de démarrage pour vérifier la connexion entre l'installation de votre client et le dispositif. Cet exemple présente les grilles de données d'entreprise.

Objectifs d'apprentissage

- Procédure de création des grilles de données dans l'interface utilisateur.
- Description du développement d'une application client dans les langages de programmation Java ou .NET. Apprenez à interopérer entre les langages de programmation en créant une grille de données d'entreprise.
- Exécution de l'application client pour insérer des données dans la grille de données.
- Surveillance des grilles de données avec la console Web.

Durée

60 minutes

| Suivant >

Leçon 1.1 du tutoriel d'initialisation : Définition de grilles de données

Vous pouvez définir des grilles de données dans l'interface utilisateur. Pour les besoins de ce tutoriel, créez une grille de données que vous appellerez `my_simple_data_grid`.

Tâches associées:

Création de grilles de données simples

Création de grilles de données simples

Une grille de données simple vous permet d'effectuer des opérations de création, de récupération, de mise à jour et de suppression.

- Le dispositif doit être initialisé et configuré. Pour plus d'informations, voir Installation de WebSphere DataPower XC10 Appliance.

Créez la grille de données simple. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Grille de données simple**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données simple grille de données à créer. Pour les besoins de ce tutoriel, créez une grille de données que vous appellerez `my_simple_data_grid`.

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à :

- Créer une grille de données simple dans l'interface utilisateur.

Module 2 du guide d'initiation : Création d'une application client

Ecrire des applications client pour insérer, mettre à jour, supprimer et extraire des données depuis la grille de données. Vous pouvez utiliser l'exemple d'application pour apprendre à créer une application pour votre environnement.

Objectifs d'apprentissage

A la fin de ce module, vous saurez :

- Java Développer une application client Java
- .NET Développer une application client .NET
- Développer une application de grille de données d'entreprise

Les leçons dans ce module

Leçon 2.1 du tutoriel d'initiation : Création d'une application client Java

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple d'initiation inclut une application client Java que vous pouvez utiliser pour en savoir plus sur la création de votre propre application client.

Leçon 2.2 du tutoriel d'initiation : Création d'une application .NET

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple du tutoriel d'initiation contient une application client .NET que vous pouvez utiliser pour apprendre à créer votre propre application client.

Leçon 2.3 : Création d'une application de grille de données

Pour créer une application de grille de données dans laquelle les clients Java et .NET peuvent mettre à jour la même grille de données, vous devez rendre les classes compatibles. Dans les exemples d'application du tutoriel d'initiation, l'exemple d'application .NET a des alias pour correspondre aux valeurs par défaut Java.

Leçon 2.1 du tutoriel d'initiation : Création d'une application client Java

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple d'initiation inclut une application client Java que vous pouvez utiliser pour en savoir plus sur la création de votre propre application client.

Le fichier `Client.java` dans le répertoire *racine_install_wxs*`/ObjectGrid/gettingstarted/client/src/` est le programme client qui montre comment se connecter à un serveur de catalogue, obtenir l'instance ObjectGrid et utiliser l'API ObjectMap. L'API ObjectMap stocke les données comme paires clé-valeur et elle est idéale pour la mise en cache d'objets qui n'ont aucune relation. Les étapes suivantes présentent le contenu du fichier `Client.java`.

1. Connectez-vous au service de catalogue en obtenant une instance ClientClusterContext.

   Pour établir la connexion au serveur de catalogues, utilisez la méthode connect de l'API ObjectGridManager. Le fragment de code suivant montre comment se connecter à un serveur de catalogue et obtenir une instance ClientClusterContext :

   ```
   ClientClusterContext ccc =
   ObjectGridManagerFactory.getObjectGridManager().connect(cep, null, null);
   ```

   La méthode connect tente de se connecter à chaque appliance de la liste jusqu'à ce qu'elle puisse établir une connexion. Un basculement automatique est effectué si l'un des autres dispositifs ne répond pas.

   Si les connexions aux serveurs de catalogue aboutissent, la méthode connect retourne une instance ClientClusterContext. L'instance ClientClusterContext est nécessaire pour obtenir la grille d'objets ObjectGrid depuis l'API ObjectGridManager.
2. Obtenez une instance ObjectGrid.

   Pour obtenir une instance ObjectGrid, utilisez la méthode getObjectGrid de l'API ObjectGridManager. La méthode getObjectGrid requiert l'instance ClientClusterContext et le nom de l'instance de grille de données. L'instance ClientClusterContext est obtenue pendant la connexion au serveur de catalogue. Le nom de l'instance de grille de données est le nom de la grille de données simple que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir la grille de données en appelant la méthode getObjectGrid de l'API ObjectGridManager.

   ```
   ObjectGrid grid = ObjectGridManagerFactory.getObjectGridManager().getObjectGrid(ccc,
   "my_simple_data_grid");
   ```

3. Définissez les informations d'identification de sécurité nécessaires.

   Créez une configuration de sécurité du client avec un nom d'utilisateur et un mot de passe que vous fournissez à l'application. Le nom d'utilisateur et le mot de passe que vous utilisez doivent avoir l'autorisation d'accéder à la grille de données sur le dispositif. Pour plus d'informations sur la création d'un utilisateur autorisé, voir Gestion des utilisateurs et des groupes.

   ```
                   // Creates a ClientSecurityConfiguration object using the specified
   file
                   ClientSecurityConfiguration clientSC =
   ClientSecurityConfigurationFactory.getClientSecurityConfiguration();
                   clientSC.setSecurityEnabled(true);
                   // Creates a CredentialGenerator using the passed-in user and
   password.
                   CredentialGenerator credGen = new
   UserPasswordCredentialGenerator(username,password);
                   clientSC.setCredentialGenerator(credGen);
                   return clientSC;
   ```

4. Obtenez une instance Session.

   Vous pouvez obtenir une session de l'instance ObjectGrid obtenue. Une instance Session est indispensable pour obtenir l'instance ObjectMap et pour effectuer une démarcation de transaction. Le fragment de code suivant montre comment obtenir une instance Session en appelant la méthode getSession de l'API ObjectGrid.

```
Session sess = grid.getSession();
```

5. Obtenez une instance ObjectMap.

   Après avoir obtenu une instance Session, vous pouvez obtenir une instance ObjectMap depuis une instance Session en appelant la méthode getMap de l'API Session. Le nom d'instance de mappe que vous envoyez à la méthode getMap porte le nom de la grille de données que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir ObjectMap en appelant la méthode getMap de l'API Session.

   ```
   ObjectMap map1 = sess.getMap("my_simple_data_grid");
   ```

   L'exemple précédent utilise l'instance de mappe par défaut qui est nommé d'après la grille de données. Vous pouvez également indiquer un nouveau nom de mappe, comme dans les exemples suivants :

   ```
   ObjectMap map2 = sess.getMap("my_simple_data_grid.CT.P");
   ObjectMap map3 = sess.getMap("my_new_map.NONE");
   ```

   La mappe `my_simple_data_grid.CT.P` est une mappe qui utilise l'expulsion en fonction de l'heure de création et le verrouillage pessimiste. La mappe `my_new_map.NONE` ne dispose pas de paramètres d'expulsion ou de verrouillage. Pour plus d'informations, voir Options de configuration de mappe dynamique.

6. Utilisez les méthodes ObjectMap.

   Une fois une instance ObjectMap obtenue, vous pouvez utiliser l'API ObjectMap. N'oubliez pas que l'interface ObjectMap est une mappe transactionnelle et qu'elle requiert une démarcation de transaction à l'aide des méthodes begin et commit de l'API Session. Faute de démarcation de transaction explicite, les opérations ObjectMap s'exécutent avec des transactions de validation automatique.

   La clé que vous utilisez peut avoir n'importe quel type Java existant, tel que java.lang.String ou Entier. Les valeurs peuvent correspondre à n'importe quel type d'objet sérialisable.

   - Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une transaction de validation automatique.

     ```
     map1.insert(key1, value1);
     ```

   - Vous pouvez exécuter une transaction sur une seule partition à la fois ou sur plusieurs partitions. Pour exécuter une transaction sur une seule partition, utilisez une transaction de validation en une phase :

     ```
     sess.setTxCommitProtocol(TxCommitProtocol.ONEPHASE);
     sess.begin();
     map1.insert(k, v);
     sess.commit();
     ```

     Pour exécuter une transaction sur plusieurs partitions, utilisez une transaction de validation en deux phases :

     ```
     sess.setTxCommitProtocol(TxCommitProtocol.TWOPHASE);
     sess.begin();
     map1.insert(k, v);
     sess.commit();
     ```

7. Facultatif : Fermez la session. Une fois toutes les opérations Session et ObjectMap terminées, fermez la session à l'aide de la méthode Session.close(). Cette méthode renvoie les ressources qui étaient utilisées par la session.

   ```
   sess.close();
   ```

   Par conséquent, les appels suivants de la méthode getSession() sont plus rapides, et moins d'objets Session se trouvent dans le segment.

Concepts associés:

Développement d'applications de grilles de données avec des API Java

Tâches associées:

Accès à la documentation des API Java

Information associée:

Documentation sur les API

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à créer une application client simple pour effectuer des opérations de grille de données.

< Précédent | **Suivant >**

Leçon 2.2 du tutoriel d'initiation : Création d'une application .NET

Pour pouvoir insérer, supprimer, mettre à jour et extraire des données dans votre grille de données, vous devez écrire une application client. L'exemple du tutoriel d'initiation contient une application client .NET que vous pouvez utiliser pour apprendre à créer votre propre application client.

- Vous devez avoir installé WebSphere eXtreme Scale Client pour .NET. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client for .NET.
- Le fichier de projet de l'exemple fonctionne avec Microsoft Visual Studio 2010 et les versions suivantes. Si vous utilisez une version précédente de Microsoft Visual Studio, vous devez créer votre propre fichier de projet.

Vous pouvez utiliser l'exemple d'application .NET du tutoriel pour :

- Vérifier que vous avez installé correctement WebSphere eXtreme Scale Client for .NET.
- Apprendre à écrire des applications pour le client .NET qui communiquent avec la grille de données pour créer des applications personnalisées. L'exemple montre comment se connecter à une grille de données sur un serveur de catalogue distant. Le mode interactif montre comment exécuter des transactions manuelles en utilisant la mappe GridMapPessimisticTx. Le mode de ligne de commande montre des transactions validées automatiquement avec la mappe GridMapPessimisticAutoTx.
- Apprendre à interagir avec l'exemple du tutoriel d'initiation iJava™. Les deux exemples d'applications stockent les éléments dans la grille de données avec les paires TestKey/TestValue. L'exemple .NET contient les attributs ClassAlias et FieldAlias pour créer des identificateurs uniques pour la sérialisation et la désérialisation. Si une opération d'insertion de clé est exécutée depuis l'application client Java, le client .NET peut obtenir la valeur en exécutant une opération get sur la clé insérée.

L'exemple d'application du tutoriel d'initiation .NET a les limitations suivantes :

- Seul le verrouillage pessimiste est pris en charge.
- Les opérations de validation en deux phases sont prises en charge. Vous pouvez valider les opérations dans une seule partition. Si vous exécutez une validation qui implique plusieurs partitions, une exception MultiplePartitionWriteException est générée.
- L'exemple ne prend pas en charge les valeurs null. L'API .NET autorise les valeurs null, mais vous devez utiliser des types pouvant avoir la valeur "null".

Le fichier de projet `SimpleClient.csproj` se trouve dans le répertoire *net_client_home*`/sample/SimpleClient`. Ce fichier de projet est le programme client qui montre comment connecter un serveur de catalogue, obtenir l'instance ObjectGrid et utiliser l'API ObjectMap. L'API ObjectMap stocke les données comme paires clé-valeur et elle est idéale pour la mise en cache d'objets qui n'ont aucune relation. Les étapes suivantes contiennent des informations sur le contenu de clé du fichier `SimpleClient.csproj`. Vous pouvez également consulter le fichier de projet plus en détail dans Microsoft Visual Studio.

Le tutoriel montre comment utiliser IGridMapPessimisticTx qui est la mappe de transactions manuelles utilisée lorsque l'application est exécutée en mode interactif. Si vous utilisez l'application en mode de ligne de commande, la mappe IGridMapPessimisticAutoTx est utilisée.

1. Connectez-vous au service de catalogue en obtenant une instance IClientConnectionContext.

   Pour vous connecter au serveur de catalogue, utilisez la méthode Connect de l'API IGridManager.

   ```
   IGridManager gm = GridManagerFactory.GetGridManager( );
   ICatalogDomainInfo cdi = gm.CatalogDomainManager.CreateCatalogDomainInfo( endpoint
   );
   ccc = gm.Connect( cdi, "SimpleClient.properties" );
   ```

   Si la connexion au serveur de catalogue aboutit, la méthode Connect retourne une instance IClientConnectionContext. L'instance IClientConnectionContext est nécessaire pour obtenir la grille de données de l'API IGridManager.

2. Obtenez une instance ObjectGrid.

   Pour obtenir une instance ObjectGrid, utilisez la méthode GetGrid de l'API IGridManager. La méthode GetGrid nécessite l'instance IClientConnectionContext et le nom de l'instance de grille de données. L'instance IClientConnectionContext est obtenue pendant la connexion au serveur de catalogue. Le nom de l'instance de grille de commande est la grille définie dans le fichier `objectgrid.xml`.

   ```
   grid = gm.GetGrid( ccc, gridName );
   ```

3. Obtenez une instance de mappe.

   Vous pouvez obtenir une instance de mappe en appelant la méthode GetGridMapPessimisticTx de l'API IGrid. Envoyez le nom de la mappe comme paramètre à la méthode GetGridMapPessimisticTx pour obtenir l'instance de mappe.

   ```
   pessMap = grid.GetGridMapPessimisticTx<Object, Object>( mapName );
   ```

4. Utilisez les méthodes IGridMapPessimisticTx.

   Une fois une instance de mappe obtenue, vous pouvez utiliser l'API IGridMapPessimisticTx.

   Le fragment de code suivant montre comment utiliser l'API IGridMapPessimisticTx.

   - Pour lancer une transaction avec l'API IGridMapPessimisticTx, vous devez appeler la méthode map.Transaction.Begin(). Cette méthode lance une nouvelle transaction dans laquelle vous pouvez exécuter des opérations.

     ```
     case "begin":
        map.Transaction.Begin( );
        return 0;
     ```

   - La méthode add insère une nouvelle paire clé/valeur. Si la clé existe, une exception est émise.

     ```
      case "a":
        if( key == null ) throw new MissingParameterException( "key" );
        if( value == null ) throw new MissingParameterException( "value" );
        map.Add( key, value );
        Console.WriteLine( "SUCCESS: Added key '{0}' with value '{1}',
        partitionId={2}", key, value, partitionId );
        return 0;
     ```

   - La méthode put insère ou met à jour une paire clé/valeur.

     ```
     case "p":
       if( key == null ) throw new MissingParameterException( "key" );
       if( value == null ) throw new MissingParameterException( "value" );
       map.Put( key, value );
       Console.WriteLine( "SUCCESS: Put key '{0}' with value '{1}',
       partitionId={2}", key, value, partitionId );
       return 0;
     ```

   - La méthode replace remplace une paire clé/valeur existante. Si l'élément n'est pas présent, une exception est émise.

     ```
     case "r":
       if( key == null ) throw new MissingParameterException( "key" );
       if( value == null ) throw new MissingParameterException( "value" );
       map.Replace( key, value );
       Console.WriteLine( "SUCCESS: Replaced key '{0}' with value '{1}',
       partitionId={2}", key, value, partitionId );
       return 0;
     ```

   - La méthode remove supprime une paire clé/valeur.

     ```
     case "d":
       if( key == null ) throw new MissingParameterException( "key" );
       map.Remove( key );
       Console.WriteLine( "SUCCESS: Deleted value with key '{0}',
       partitionId={1}", key, partitionId );
              return 0;
     ```

   - La méthode get extrait la valeur de la clé.

     ```
      case "g":
       if( key == null ) throw new MissingParameterException( "key" );
       value = ( TestValue )map.Get( key );
       if ( value != null )
       {
     ```

```
    Console.WriteLine( "SUCCESS: Value is '{0}',
  partitionId={1}", value, partitionId );
  }
  else
  {
    Console.WriteLine( "FAILED: Key not found" );
  }
  return 0;
```

- Si vous voulez annuler les opérations que vous avez exécutées dans l'opération avant la validation, utilisez la méthode rollback.

```
case "rollback":
  map.Transaction.Rollback( );
  return 0;
```

- La méthode commit valide les opérations exécutées dans la transaction.

```
case "commit":
  map.Transaction.Commit( );
  return 0;
```

Tâches associées:

.NET Configuration de l'environnement de développement .NET
.NET Accès à la documentation des API WebSphere eXtreme Scale Client for .NET

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à créer une application .NET simple pour exécuter des opérations de grille de données.

< Précédent | **Suivant >**

Leçon 2.3 : Création d'une application de grille de données

Pour créer une application de grille de données dans laquelle les clients Java™ et .NET peuvent mettre à jour la même grille de données, vous devez rendre les classes compatibles. Dans les exemples d'application du tutoriel d'initiation, l'exemple d'application .NET a des alias pour correspondre aux valeurs par défaut Java.

Ajoutez des alias de classe et des attributs d'alias de zone à l'application .NET. Vous pouvez ajouter les alias de classe à l'application .NET, l'application Java ou aux deux applications. L'exemple .NET a des alias qui correspondent aux valeurs par défaut Java. Par conséquent, l'application Java n'a pas besoin d'alias. Les fichiers `TestKey.cs` et `TestValue.cs` se trouvent dans le répertoire *net_client_home*`/sample/SimpleClient`.

Figure 1. Attribut d'alias de classe dans le fichier TestKey.cs

```
[ClassAlias( "com.ibm.websphere.xs.sample.gettingstarted.model.TestKey" )]
```

Figure 2. Attribut d'alias de classe dans le fichier TestValue.cs

```
[ClassAlias( "com.ibm.websphere.xs.sample.gettingstarted.model.TestValue" )]
```

Concepts associés:

Annotations ClassAlias et FieldAlias

Tâches associées:

Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET

Point de contrôle de la leçon

Vous avez ajouté des attributs à l'application d'initiation .NET. Par conséquent, vous pouvez utiliser l'application d'initiation Java en créant une grille de données d'entreprise.

Module 3 : Exécution de l'exemple d'application dans la grille de données

Vous pouvez exécuter l'exemple d'application client sur la grille de données de votre dispositif.

En revanche, la manière d'exécuter cette exemple d'application client varie selon qu'il s'agit de la version Java ou de la version .NET.

Objectifs d'apprentissage

A la fin de ce module, vous saurez :

- Java Exécuter l'exemple d'application client Java de l'initiation
- .NET Exécuter l'exemple d'application client .NET.

Vous pouvez exécuter les exemples d'application Java et .NET séparément, mais vous pouvez également les exécuter simultanément sur la même grille de données. Par exemple, vous pouvez insérer une valeur dans la grille de données avec l'application .NET, puis obtenir cette valeur avec l'application Java. Dans ce scénario, vous exécutez une grille de données d'entreprise.

Les leçons dans ce module

Leçon 3.1 du tutoriel d'initiation : Exécution de l'exemple d'application Java

Procédez comme suit pour exécuter un client Java pour interagir avec la grille de données.

Leçon 3.2 du tutoriel d'initiation : Exécution de l'exemple d'application .NET

Procédez comme suit pour exécuter une application WebSphere eXtreme Scale Client for .NET pour interagir avec la grille de données. Le serveur de catalogue, le serveur de conteneur et le client s'exécutent tous sur un serveur unique dans cet exemple.

< Précédent | **Suivant >**

Leçon 3.1 du tutoriel d'initiation : Exécution de l'exemple d'application Java

Procédez comme suit pour exécuter un client Java pour interagir avec la grille de données.

Modifiez le fichier *racine_install_wxs*`/ObjectGrid/gettingstarted/env.bat|sh`. Ce fichier est automatiquement appelé par le client. Il contient les informations suivantes :

```
SET CATALOGSERVER_HOST=<nom_hôte_xc10>
SET CATALOGSERVER_PORT=2809
SET GRID_NAME=ma_grille_données_simple
SET MAP_NAME=my_map.P
```

Pour la connexion à une grille de données sur le dispositif, vous devez mettre à jour les propriétés **CATALOGSERVER_HOST** et **CATALOGSERVER_PORT**. Le serveur de catalogue à spécifier est affiché dans la page d'interface utilisateur de la grille de données simple que vous avez créée. Cliquez sur **Grille de données** > **Grille de données simple** > ***ma_grille_données_simple*** et utilisez les valeurs dans la zone **Services de catalogue**. La valeur de la propriété **GRID_NAME** doit correspondre au nom de la grille de données que vous avez créée. La propriété **MAP_NAME** crée une mappe sans durée de vie (TTL) et utilisant le verrouillage pessimiste. Pour plus d'informations sur le nommage des mappes dynamiques, voir Options de configuration de mappe dynamique.

- Exécutez le client en mode interactif. Dans la fenêtre de ligne de commande, exécutez l'une des commandes suivantes :
  - UNIX Linux `./runclient.sh`
  - Windows `runclient.bat`

  1. Démarrez une transaction. Vous pouvez utiliser une opération de validation en une phase ou deux phases pour la transaction. Avec la validation en une phase, la transaction doit écrire dans une seule partition. Si pendant la transaction vous insérez des clés dans différentes partitions, la validation de la transaction échoue. Vous pouvez utiliser la validation en deux phases pour écrire dans plusieurs partitions au cours d'une même transaction.
     - Démarrez une transaction avec validation en une phase.

       ```
       begin
       ```

     - Démarrez une transaction avec validation en deux phases.

       ```
       begin2pc
       ```

  2. Insérez une valeur.

     ```
     > i key1 helloWorld
     SUCCESS: Inserted TestValue [value=helloWorld] with key TestKey [key=key1], p
     art
     itionId=6
     ```

  3. Extrayez une valeur que vous avez insérée.

     ```
     > g key1
     Value is TestValue [value=helloWorld], partitionId=6
     ```

  4. Mettez à jour une valeur.

     ```
     > u key1 goodbyeWorld
     SUCCESS: Updated key TestKey [key=key1] with value TestValue [value=goodbyeWo
     rld
     ], partitionId=6
     ```

  5. Annulez la transaction. Lorsque vous annulez la transaction, toutes les opérations associées à la transaction sont annulées.

     ```
     > rollback
     ```

6. Pour tester l'opération d'annulation, essayez d'obtenir de nouveau la clé. Comme vous avez annulé la transaction, la clé n'existe pas :

```
> g key1
```

7. Insérez une valeur.

```
> i key1 helloWorld
SUCCESS: Inserted TestValue [value=helloWorld] with key TestKey [key=key1], p
art
itionId=6
```

8. Validez la valeur. Après avoir validé la transaction, vous ne pouvez pas annuler les modifications.

```
> commit
```

9. Supprimez une valeur que vous avez insérée.

```
> d key1
SUCCESS: Deleted value with key TestKey [key=key1], partitionId=6
```

10. Insérez des entrées de test. Par exemple, pour insérer 1000 clés et valeurs numérotées de 0 à 999, utilisez la commande suivante :

```
> n 1000
```

- Exécutez le client en mode de ligne de commande. Le mode de ligne de commande peut être utile si vous voulez écrire un script pour exécuter l'application client. Vous pouvez exécuter les mêmes commandes que celles que vous exécutez en mode interactif. Voici un exemple de syntaxe pour le mode de ligne de commande :
  - UNIX Linux

```
./runclient.sh i "key1" "helloWorld"
```

  - Windows

```
runclient.bat i "key1" "helloWorld"
```

Point de contrôle de la leçon

Points étudiés

Dans cette leçon, vous avez appris à :

- Exécuter l'exemple d'application client Java pour insérer, obtenir, mettre à jour et supprimer des données de la grille de données.

< Précédent | **Suivant >**

Leçon 3.2 du tutoriel d'initiation : Exécution de l'exemple d'application .NET

Procédez comme suit pour exécuter une application WebSphere eXtreme Scale Client for .NET pour interagir avec la grille de données. Le serveur de catalogue, le serveur de conteneur et le client s'exécutent tous sur un serveur unique dans cet exemple.

WebSphere eXtreme Scale Client for .NET ne prend en charge les validations en une seule phase. Par conséquent, si vous tentez d'insérer plusieurs valeurs dans une même transaction, une exception risque de se produire, car les valeurs sont placées dans des partitions différentes. Pour empêcher l'occurrence de ces exceptions lorsque vous exécutez l'exemple, vous pouvez changer le fichier XML descripteur de règle de déploiement pour utiliser une seule partition. Pour plus d'informations sur la mise à jour du nombre de partitions, voir Leçon 1.1 du tutoriel d'initialisation : Définition de grilles de données.

Vous pouvez exécuter l'exemple d'application en mode interactif ou de ligne de commande. En mode interactif, l'application exécute les transactions de grille de données manuelles avec l'API IGridMapPessimisticTx. Le mode de ligne de commande exécute les transactions de grille de données automatiques avec l'API IGridMapPessimisticAutoTx.

Vous pouvez exécuter l'exemple en mode interactif ou de ligne de commande :

- Exécutez l'exemple d'application client en mode interactif.
  1. Exécutez l'application client simple. Le fichier se trouve dans le répertoire *net_client_home*\gettingstarted\bin\. Pour exécuter l'exemple en mode interactif, exécutez la commande suivante :

     ```
     SimpleClient.exe -i [-h <nom_hôte:port>]  [-g <nom_grille>] [-m <nom_mappe>]
     ```

     -h <nom_hôte:port>

     Indique le nom d'hôte et le port du serveur de catalogue auquel vous voulez vous connecter. Le serveur de catalogue à spécifier est affiché dans la page d'interface utilisateur de la grille de données simple que vous avez créée. Cliquez sur **Grille de données** > **Grille de données simple** > ***ma_grille_données_simple*** et utilisez les valeurs dans la zone **Services de catalogue**.

     -g <nom_grille>

     Indique le nom de la grille de données à utiliser. Vous devez utiliser une mappe avec stratégie de verrouillage pessimiste. Pour ce tutoriel, utilisez `ma_grille_données_simple` ou le nom que vous avez indiqué lorsque vous avez créé la grille de données dans le dispositif. Si vous n'indiquez pas de nom, la grille de données `Grid` est utilisée.

     -m <nom_mappe>

     Indique le nom de la mappe à utiliser. Pour ce tutoriel, indiquez `ma_mappe.NONE.P`. Si vous n'indiquez pas de valeur, une erreur se produit. Pour plus d'informations concernant les noms de mappe, voir Options de configuration de mappe dynamique.

     Si vous exécutez l'application sans paramètre, l'aide de l'application s'affiche.

  2. Affichez la liste des commandes disponibles.

     ```
     Enter a command: help
     This program executes simple CRUD operations on a map.
            a - Adds a value with the specified key. If the key already exists,
            DuplicateKeyException is thrown
            p - Adds a value with the specified key, replacing the entry if it
            already exists
            r - Replaces the value of the specified key. If the key does not exis
     t,
            a CacheKeyNotFound exception is thrown
            g - Retrieve and display the value of the specified key
            d - Deletes the key
            gp - Gets the partition id for the key
            ck - Checks if the map contains the key
            h - Display help
            begin - Begin manual transaction
            commit - Commit transactions
     ```

```
        rollback - Rollback transactions
        exit - Exit program
```

3. Démarrez la transaction. Vous devez démarrer une transaction pour pouvoir exécuter des commandes sur la grille de données. Si vous ne démarrez pas la transaction, une exception NoActiveTransacationException se produit.

```
Enter a command: begin
```

4. Ajoutez des données à la grille.

```
Enter a command: a key1 value1
SUCCESS: Added 'TestKey [key=key1]' with value 'TestValue [value=value1]',
partitionId=6
```

5. Recherchez et affichez la valeur.

```
Enter a command: g key1
SUCCESS: Value is 'TestValue [value=value1]', partitionId=6
```

   Dans cet exemple, `value1` est retourné.

6. Mettez à jour la clé. Utilisez la commande put qui ajoute une valeur avec la clé définie en remplaçant la valeur existante éventuelle.

```
Enter a command: p key1 value2
SUCCESS: Put key 'TestKey [key=key1]' with value 'TestValue [value=value2]',
 partitionId=6
Enter a command: g key1
SUCCESS: Value is 'TestValue [value=value2]', partitionId=6
```

7. Remplacez la clé. La commande remplace la valeur par la clé définie. Si la clé n'existe pas, une exception CacheKeyException est émise.

```
Enter a command: r key1 value3
SUCCESS: Replaced key 'TestKey [key=key1]' with value 'TestValue [value=value
3]'
, partitionId=6
```

8. Annulez la transaction et essayez d'afficher de nouveau la clé de valeur. Vous pouvez annuler la transaction à tout moment avant la validation.

```
Enter a command: rollback
Enter a command: begin
Enter a command: g key1
FAILED: Key not found
```

   Lorsque vous exécutez la commande **get**, vous obtenez une erreur signalant que la clé est introuvable.

9. Validez une clé et une valeur dans la grille de données.

```
Enter a command: begin
Enter a command: a key2 value2
SUCCESS: Added 'TestKey [key=key2]' with value 'TestValue [value=value2]',
partitionId=7
Enter a command: commit
```

10. Obtenez un ID de partition pour une clé.

```
Enter a command: begin
Enter a command: gp key2
SUCCESS: partitionId=7
```

11. Recherchez des clés dans la mappe.

```
Enter a command: ck key2
```

```
SUCCESS: The map contains key 'TestKey [key=key2]'
Enter a command: ck key3
SUCCESS: The map does NOT contain key 'TestKey [key=key3]'
```

12. Supprimez la clé et quittez.

```
Enter a command: begin
Enter a command: d key2
SUCCESS: Deleted value with key 'TestKey [key=key2]', partitionId=7
Enter a command: commit
Enter a command: exit
```

- Exécutez le client en mode de ligne de commande. Le mode de ligne de commande exécute les transactions de grille de données automatiques avec l'API IGridMapPessimisticAutoTx. Pour utiliser ce mode, transmettez l'action sur la ligne de commande. Le mode de ligne de commande peut être utile si vous voulez écrire un script pour exécuter l'application client. Vous pouvez exécuter les mêmes commandes que celles que vous exécutez en mode interactif. Voici un exemple de syntaxe pour le mode de ligne de commande :

```
SimpleClient [-h <hôte:port>] [-g <nom_grille>] [-m <nom_mappe>] <a | p | r | g |
d> <clé> [<valeur>]
```

Tâches associées:

.NET Développement d'applications de grille de données avec les API .NET
Accès à la documentation des API WebSphere eXtreme Scale Client for .NET

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à :

- Exécuter l'exemple d'application .NET pour insérer, obtenir, mettre à jour et supprimer des objets de la grille de données.

< Précédent | **Suivant >**

< Précédent

Leçon 4 du tutoriel du guide de démarrage : Surveillance de l'environnement

Vous pouvez utiliser l'utilitaire **xscmd** et les outils de la console Web pour surveiller votre environnement de grille de données.

Tâches associées:

Surveillance des grilles de données dans l'interface utilisateur

Surveillance des grilles de données dans l'interface utilisateur

L'interface utilisateur vous permet de visualiser les performances globales des grilles des données présentes dans votre environnement. Sa section dédiée à la surveillance offre une vue globale de toutes les grilles de données du dispositif, une vue globale de chaque grille, et des rapports détaillés sur les grilles individuelles.

Pour plus d'informations concernant la surveillance des grilles de données dans l'interface utilisateur, voir Surveillance des grilles de données dans l'interface utilisateur.

Surveillance avec l'utilitaire `xscmd`

1. Facultatif : Si l'authentification de client est activée : Sur l'installation client, ouvrez une fenêtre de ligne de commande. Sur la ligne de commande, définissez les variables d'environnement appropriées.
2. Accédez au répertoire *rép_base_wxs*`/bin`.

   `cd` *rép_base_wxs*`/bin`

3. Plusieurs commandes vous permettent d'afficher des informations concernant votre environnement.
   - Afficher tous les serveurs de conteneur en ligne pour la grille de données de la grille et le groupe de mappes mapSet :

     ```
     xscmd -c showPlacement -g Grid -ms mapSet
     ```

   - Afficher les informations de routage de la grille de données :

     ```
     xscmd -c routetable -g Grid
     ```

   - Afficher le nombre d'entrées de mappe dans la grille de données :

     ```
     xscmd -c showMapSizes -g Grid -ms mapSet
     ```

Point de contrôle de la leçon

Dans cette leçon, vous avez appris à :

- surveiller les statistiques de la grille et des serveurs ;

< Précédent

Planification de l'environnement DataPower XC10 Appliance

Pour pouvoir intégrer DataPower XC10 Appliance à la topologie, l'environnement doit répondre aux conditions et à la configuration logicielle suivantes.

Configuration requise
Votre environnement doit présenter la configuration suivante pour permettre l'intégration avec WebSphere DataPower XC10 Appliance.

Conventions relatives aux répertoires
Les conventions de répertoire suivantes sont utilisées dans toute la documentation pour faire référence à des répertoires spéciaux, tels que *wxs_install_root* et *wxs_home*. Vous pouvez accéder à ces répertoires pendant plusieurs scénarios différents, y compris lors de l'installation et de l'utilisation des outils de ligne de commande.

Ports réseau
Si vous utilisez WebSphere DataPower XC10 Appliance derrière un pare-feu, vous devez activer la communication via les ports ci-dessous.

Conditions nécessaires à l'installation d'IBM WebSphere DataPower XC10 Appliance
Vous devez satisfaire les exigences en termes de matériel, logiciel, armoire et outils afin d'installer et de configurer IBM® WebSphere DataPower XC10 Appliance. Utilisez cette liste de prérequis pour planifier l'installation, la configuration et l'utilisation d'IBM WebSphere DataPower XC10 Appliance.

Spécifications et fonctions du dispositif
Le matériel de type 7199-92x prend en charge WebSphere DataPower XC10 Appliance version 2.5. Le matériel de type 9235-92x était livré avec la version 1.0 et n'est pris en charge que jusqu'à la version 2.1.

Interopérabilité du produit
WebSphere DataPower XC10 Appliance a été testé du point de vue de l'interopérabilité avec d'autres produits IBM.

.NET 2.5+ ### Remarques relatives à Microsoft .NET
Deux environnements .NET existent dans WebSphere eXtreme Scale : l'environnement de développement et l'environnement d'exécution. Ces environnements ont des exigences spécifiques.

Configuration requise

Votre environnement doit présenter la configuration suivante pour permettre l'intégration avec WebSphere DataPower XC10 Appliance.

Configuration matérielle

Pour effectuer la configuration initiale, vous devez utiliser une connexion série. La connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif WebSphere DataPower XC10 Appliance. Pour établir la connexion série, utilisez le câble série faux modem RJ45/DB-9 fourni.

Remarque : Le terminal n'est pas équipé d'un port série ; utilisez un câble adaptateur USB/Série.

Vous pouvez effectuer une installation automatisée ou à distance via un serveur de terminal connecté au réseau.

Configuration logicielle requise

Le dispositif WebSphere DataPower XC10 Appliance est fourni avec un logiciel IBM. Vous ne pouvez pas installer d'autre logiciel sur le dispositif.

Navigateurs Web requis

L'interface utilisateur prend en charge les navigateurs Web suivants :

- Mozilla Firefox, version 3.5.x et versions ultérieures
- Microsoft Internet Explorer version 7 ou ultérieure

Logiciels pris en charge

Vous pouvez utiliser les logiciels suivants avec WebSphere DataPower XC10 Appliance :

- WebSphere eXtreme Scale Client 8.5
- WebSphere eXtreme Scale Client version 8.6
- WebSphere Application Server version 6.1.0.41 ou ultérieure
- WebSphere Application Server version 7.0.0.21 ou ultérieure
- WebSphere Application Server version 8.0.0.2 ou ultérieure
- WebSphere Application Server version 8.5.0.0 ou ultérieure
- WebSphere Application Server version 8.5.5.0

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Conventions relatives aux répertoires

Les conventions de répertoire suivantes sont utilisées dans toute la documentation pour faire référence à des répertoires spéciaux, tels que *wxs_install_root* et *wxs_home*. Vous pouvez accéder à ces répertoires pendant plusieurs scénarios différents, y compris lors de l'installation et de l'utilisation des outils de ligne de commande.

racine_install_wxs

Le répertoire *wxs_install_root* est le répertoire racine où sont installés les fichiers du produit WebSphere eXtreme Scale. Le répertoire *wxs_install_root* peut être le répertoire dans lequel l'archive d'évaluation est extraite ou à partir duquel le produit est installé WebSphere eXtreme Scale.

- Exemple où la version d'essai a été extraite :

  Exemple : `/opt/IBM/WebSphere/eXtremeScale`

- Exemple lorsque WebSphere eXtreme Scale est installé dans un répertoire autonome :

  UNIX **Exemple :** `/opt/IBM/eXtremeScale`

  Windows **Exemple :** `C:\Program Files\IBM\WebSphere\eXtremeScale`

- Exemple lorsque WebSphere eXtreme Scale est intégré à WebSphere Application Server :

  Exemple : `/opt/IBM/WebSphere/AppServer`

wxs_home

Le répertoire *wxs_home* est le répertoire racine du produit, des bibliothèques, des exemples et des composants WebSphere eXtreme Scale. Ce répertoire est identique au répertoire *wxs_install_root* lorsque l'archive d'évaluation est extraite. Pour les installations autonomes, le répertoire *wxs_home* est le sous-répertoire `ObjectGrid` du répertoire *wxs_install_root*. Pour les installations qui sont intégrées à WebSphere Application Server, ce répertoire est le répertoire `optionalLibraries/ObjectGrid` du répertoire *wxs_install_root*.

- Exemple lorsque la version d'essai a été extraite :

  Exemple : `/opt/IBM/WebSphere/eXtremeScale`

- Exemple lorsque WebSphere eXtreme Scale est installé dans un répertoire autonome :

  UNIX **Exemple :** `/opt/IBM/eXtremeScale/ObjectGrid`

  Windows **Exemple :** *racine_install_wxs*`\ObjectGrid`

- Exemple lorsque WebSphere eXtreme Scale est intégré à WebSphere Application Server :

  Exemple : `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid`

was_root

Le répertoire *was_root* est le répertoire racine d'une installation WebSphere Application Server :

Exemple : `/opt/IBM/WebSphere/AppServer`

.NET **net_client_home**

Le répertoire *net_client* est le répertoire racine d'une installation client .NET.

Exemple : `C:\Program Files\IBM\WebSphere\eXtreme Scale .NET Client`

java_home

Le répertoire *java_home* est le répertoire racine d'une installation de Java™ Runtime Environment Kit (JRE).

UNIX **Exemple :** `/opt/IBM/WebSphere/eXtremeScale/java`

Windows **Exemple :** *racine_install_wxs*`\java`

rep_base_exemples

rep_base_exemples est le répertoire dans lequel vous extrayez les exemples de fichiers qui sont utilisés pour les tutoriels.

UNIX **Exemple :** *rép_base_wxs*`/samples`

Windows **Exemple :** *rép_base_wxs*`\samples`

dvd_root

dvd_root est le répertoire racine du DVD qui contient le produit.

Exemple : `dvd_root/docs/`

rep_base_utilisateur

Le répertoire *rep_base_utilisateur* est l'emplacement de stockage des fichiers utilisateur, tels que les profils de sécurité.

Windows `c:\Documents and Settings\`*user_name*

UNIX `/home/`*user_name*

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Ports réseau

Si vous utilisez WebSphere DataPower XC10 Appliance derrière un pare-feu, vous devez activer la communication via les ports ci-dessous.

Serveur de catalogue

Les ports ci-dessous sont utilisés par le serveur de catalogue. Chacun des trois premiers dispositifs ajoutés à la collectivité exécute un serveur de catalogue.
Port homologue : Utilisé pour la communication entre les serveurs de catalogue. Ce port est défini pour utiliser le numéro 6601.
Port client : Utilisé pour la communication entre les serveurs de catalogue. Ce port est défini pour utiliser le numéro 6602.
Port de service JMX : Utilisé pour les connexions JMX non SSL, notamment pour l'utilitaire xscmd. Ce port est défini pour utiliser le numéro 1099.
Port de connecteur JMX : Utilisé pour les connexions JMX, notamment pour l'utilitaire xscmd. Ce port est défini pour utiliser le numéro 1100 et sélectionne un port entre 7100 et 7116.
Port d'écoute ORB : Utilisé pour la communication entre le client et les serveurs de grille de données. Ce port est défini pour utiliser le numéro 2809.
Port d'écoute CSIv2 : Utilisé pour la communication sécurisée entre le client et les serveurs de grille de données. Ce port est défini pour utiliser le numéro 7499.
Port SNMP : Utilisé pour la surveillance SNMP. Ce port est défini pour utiliser le numéro 161.

Remarque : WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Les ports suivants sont nécessaires à la communication entre les serveurs de conteneur :

Ports de groupe central DCS : Utilisés pour la communication DCS interne de WebSphere DataPower XC10 Appliance. Ce port utilise un port compris entre 6700 et 6716.
Ports d'écoute ORB : Utilisés pour la communication entre le client et les serveurs de grille de données. Ce port utilise un port compris entre 6800 et 6816.
Port d'écoute CSIv2 : Utilisé pour les communications sécurisées entre le client et les serveurs de grille de données. Ce port utilise un port compris entre 7500 et 7516.

Remarque : Si les dispositifs appartiennent à une collectivité et ne sont pas tous situés du même côté du pare-feu, alors les ports DCS, ORB et CSIv2 (serveurs de conteneur), ainsi que les ports 6601 et 6602 (serveurs de catalogue) doivent être ouverts aux communications bidirectionnelles.

Ports de navigateur

Ouvrez les ports 80 et 443 si vous voulez utiliser la console du dispositif ou des services REST. Certaines options de configuration peuvent échouer si ces ports ne sont pas ouverts sur tous les dispositifs de la collectivité.

Serveurs client

WebSphere eXtreme Scale Client utilise un port d'écoute. Ce port est défini dans le fichier de propriétés du client, dans le répertoire *wxs_client_root*\properties.

Application d'identification et de résolution des problèmes

Ouvrez le port 9060 pour l'application d'identification et de résolution des problèmes.

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Conditions nécessaires à l'installation d'IBM WebSphere DataPower XC10 Appliance

Vous devez satisfaire les exigences en termes de matériel, logiciel, armoire et outils afin d'installer et de configurer IBM® WebSphere DataPower XC10 Appliance. Utilisez cette liste de prérequis pour planifier l'installation, la configuration et l'utilisation d'IBM WebSphere DataPower XC10 Appliance.

Compétences requises

Pour configurer et gérer le dispositif, vous devez disposer de compétences en matière d'administration de réseau.

Informations requises

Rassemblez les informations suivantes pour définir la configuration de base de votre système DataPower XC10 Appliance.

- Utilisation de ports Ethernet 1 gigabits ou utilisation de ports Ethernet 10 gigabits pour votre grille de données. Vous devez utiliser tous les ports 1 gigabits ou tous les ports 10 gigabits. Connectez le port de gestion à `MGMT0`.
- Adresse IP et masque de sous-réseau de l'interface Ethernet pour accès à la gestion du dispositif (MGMT).
- Adresse IP des passerelles par défaut (routeurs) prenant en charge les sous-réseaux des interfaces Ethernet.
- Adresse IP du serveur DNS (Domain Name System). Configurez le serveur DNS pour recherche directe et inversée.
- Paramètres de communication pour interface série : 9600.8.n.1 (9600 bauds, 8 bits de données, pas de parité, 1 bit d'arrêt).
- Informations sur le serveur de messagerie (pour configuration des notifications par courrier électronique).
- (Facultatif) Adresses IP et masque de sous-réseau des interfaces Ethernet pour les accès de service au dispositif (ETH0, ETH1 et ETH2).

Exigences relatives à l'armoire

Pour installer le système DataPower XC10 Appliance, vous devez disposer d'une armoire standard de 19 pouces (48,26 cm) avec une profondeur minimale de 25 pouces (63,50 cm) répondant aux critères suivants :

- Présence de colonnes de montage à l'arrière
- Prise en charge de montage avant et arrière

- Présence de colonnes de montage à l'arrière
- Prise en charge de montage avant et arrière

Veillez à ce que l'espace libre requis suivant soit disponible dans l'armoire et autour de celle-ci :

- Au moins 30 pouces (76,20 cm) d'espace libre derrière l'armoire
- Au moins 2 pouces (5,1 cm) au dessus et en dessous du dispositif
- Espace libre suffisant à l'avant pour câbles Ethernet et de console série

La température ambiante dans l'environnement où doit être installé le dispositif ne doit pas dépasser 104° F (40° C).

Outils et équipement

Pour installer le système DataPower XC10 Appliance, vous devez rassembler l'équipement suivant :

- Un tournevis de diamètre moyen
- Deux (2) vis pour armoire standard (fournies avec le dispositif)
- Un à quatre (1-4) câbles réseau
- Câble série ou câble USB/série PL-2303

  Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de pouvoir continuer.

  Remarque : Ne vous débarrassez pas des câbles après l'installation du dispositif. Vous en aurez peut-être besoin pour identifier des problèmes ou effectuer certaines opérations de maintenance à l'avenir.

- Console série avec un connecteur mâle DB9. Il peut s'agir d'un matériel dédié, par exemple une console VT100, ou d'un PC exécutant un émulateur comme HyperTerminal ou Minicom.

Exigences de l'utilisateur interface

Pour utiliser l'interface utilisateur Web, vous pouvez utiliser l'un des navigateurs suivants :

- Mozilla Firefox, version 3.5.x et versions ultérieures

- Microsoft Internet Explorer version 7 ou ultérieure

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Spécifications et fonctions du dispositif

Le matériel de type 7199-92x prend en charge WebSphere DataPower XC10 Appliance version 2.5. Le matériel de type 9235-92x était livré avec la version 1.0 et n'est pris en charge que jusqu'à la version 2.1.

Pour déterminer le type de dispositif, cliquez sur **Dispositif** > **Paramètres** > **Microprogramme**. Le panneau affiche le type de modèle et le numéro de série du dispositif.

Spécifications et fonctions du dispositif type 7199-92x
Utilisez les spécifications et les fonctions pour déterminer l'environnement physique nécessaire pour contenir votre dispositif.

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Spécifications et fonctions du dispositif type 7199-92x

Utilisez les spécifications et les fonctions pour déterminer l'environnement physique nécessaire pour contenir votre dispositif.

A propos des dispositifs de type 7199-92X

Le type 7199-92X WebSphere DataPower XC10 Appliance est livré avec la version 2.0 du logiciel WebSphere DataPower XC10 Appliance. Cette nouvelle version du matériel inclut des processeurs plus rapides, davantage de ports réseau et davantage de capacité de mémoire cache que les versions antérieures.

Pour déterminer le type de dispositif, cliquez sur **Dispositif** > **Paramètres** > **Microprogramme**. Le panneau affiche le type de modèle et le numéro de série du dispositif.

Spécifications

Tableau 1. Spécifications du dispositif de type 7199-92x. Récapitule les spécifications du châssis de type 7199.

Dimensions :	
	7199
Hauteur	8,89 cm (3,5 pouces)
Largeur	42,8 cm (17,25 pouces)
Profondeur	58,4 cm (23 pouces)
Poids	Maximum : 21 kg
Alimentation électrique :	
Tension sinusoïdale	50 - 60 Hz (monophasé) requis
110 Volts CA	Minimum : 100 V_{RMS} Maximum : 127 V_{RMS}
220 Volts CA	Minimum : 200 V_{RMS} Maximum : 240 V_{RMS}
Consommation	10 A pour 110 V CA, 5 A pour 220 V CA Le dispositif de type 7199 contient deux modules de 720 watts. Les deux modules d'alimentation doivent être connectés à la même source de courant pour éviter une différence dans le voltage "terre" entre les deux modules.
Environnement :	
Température ambiante	En opération : • Altitude : de 0 à 914,4 m (3000 ft.)50° à 95° F (10° à 35° C) • Altitude : de 914,4 m (3000 ft.) à 2133,6 m (7000 ft.)50° à 89,6° F (10° à 32° C) Altitude maximale : 2133,6 m (7000 ft.) Hors tension : 50° à 109,4° F (10° à 43° C) Transport : -40° à 140° F (-40° à 60° C)
Humidité	8% à 80%

Caractéristiques

Tableau 2. Options de stockage des données

Caractéristique	Description
Capacité locale	16 Go de stockage sur le système de fichiers local
Grappe de disques durs	Deux unités de disque dur SAS (Serial Attached SCSI) permutables simples de 1 To Capacité : 2 To
Grille de données	Unité Fusion DUO de 320 Go, qui inclut la mémoire cache de 240 Go

Vue avant du type 7199-92x

La vue avant montre les contrôles, les voyants et les connecteurs du dispositif de type 7199. Les modules Ethernet et les modules d'unité de disque dur peuvent être installés à partir du panneau frontal du dispositif de type 7199-92x.

Vue arrière du type 7199-92x

La vue arrière montre les composants et les voyants situés à l'arrière du dispositif. Les modules de ventilation et les modules d'alimentation sont installés depuis l'arrière du dispositif.

Configuration du réseau Ethernet

Les modules Ethernet étendent les options de connectivité réseau. Chaque dispositif a deux modules Ethernet. Les modules Ethernet sont numérotés de gauche à droite, mais si un module a moins de huit ports, il utilisera le numéro de port le plus petit de la plage.

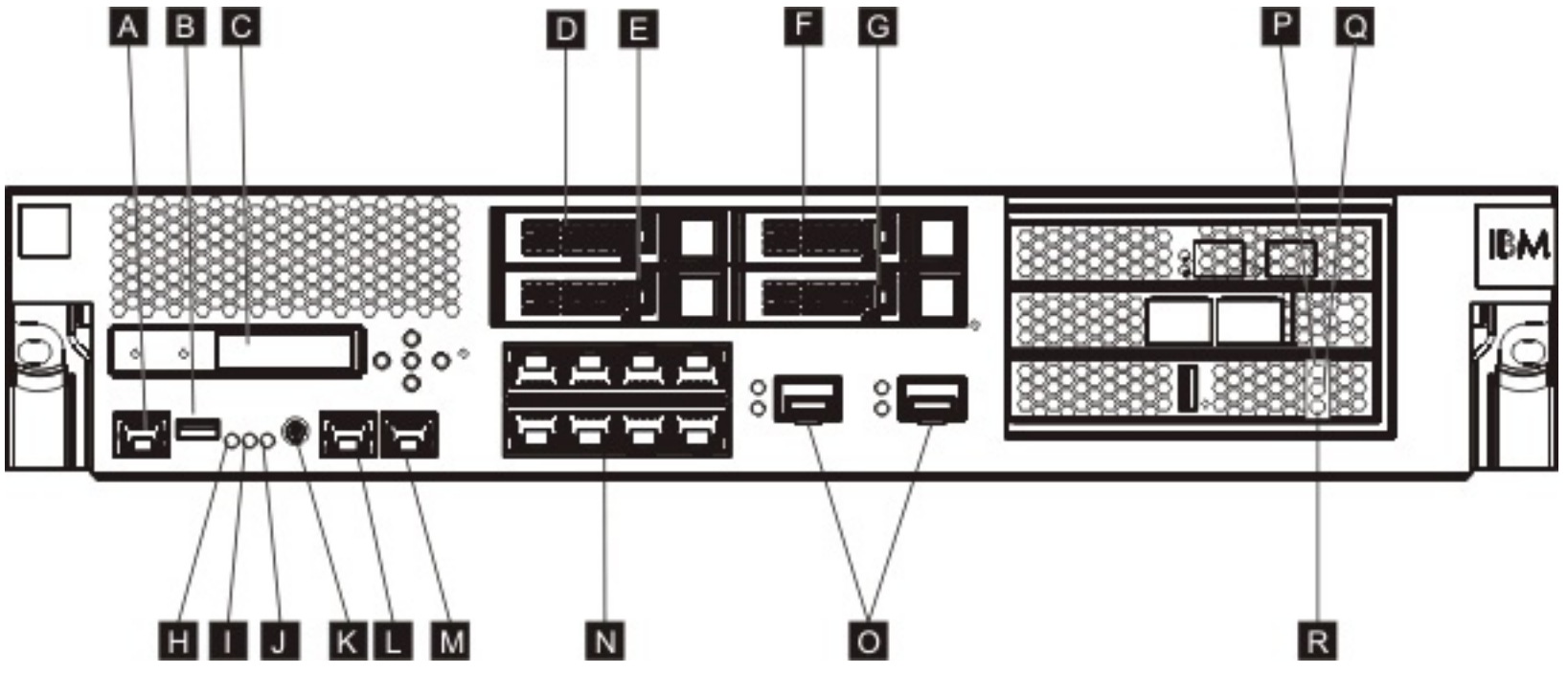
Rubrique parent : Spécifications et fonctions du dispositif

Vue avant du type 7199-92x

La vue avant montre les contrôles, les voyants et les connecteurs du dispositif de type 7199. Les modules Ethernet et les modules d'unité de disque dur peuvent être installés à partir du panneau frontal du dispositif de type 7199-92x.

Diagramme de la vue avant

Figure 1. Vue avant du type 7199-92x



Les libellés du diagramme correspondent aux composants suivants du panneau avant d'un dispositif de type 7199-92x :

A

Connecteur de la console

B

Port USB

C

Module LCM

D

Module d'unité de disque dur 2

E

Module d'unité de disque dur 0

F

Module d'unité de disque dur 3

G

Module d'unité de disque dur 1

H

Voyant d'erreur

I

Voyant d'emplacement

J

Voyant d'alimentation

K

Bouton d'alimentation

L

Connecteur Ethernet `MGT0`

M

Connecteur Ethernet `MGT1`

N

Modules Ethernet de gauche :

- eth0
- eth1

- eth2
- eth3
- eth4
- eth5
- eth6
- eth7

O

Modules Ethernet de droite :

- eth8
- eth9

P

Voyant ambre ou d'erreur du cache.

Q

Voyant jaune ou d'écriture du cache.

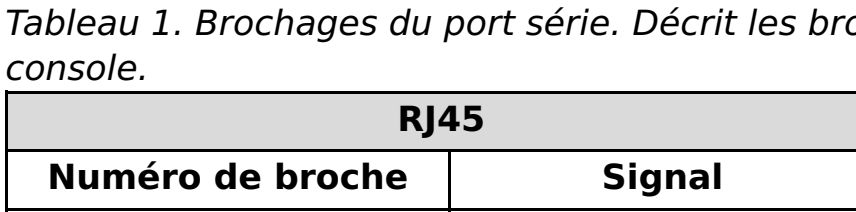
R

Voyant vert ou de lecture du cache.

Module d'affichage

Le panneau avant du dispositif a un module d'affichage à cristaux liquides (LCD) incluant un écran d'affichage LCD et cinq boutons de menu. L'écran LCD fournit des informations sur le type de modèle du dispositif ; cependant, les boutons de menu ne sont pas fonctionnels.

Figure 2. Module d'affichage



Connecteur de la console

Le panneau frontal du dispositif de type 7199 a un connecteur de console. Pour la configuration initiale, utilisez le câble série faux modem RJ45 (IS0 8877) à DB-9 (également appelé DE-9 ou EIA/TIA-562) qui est livré avec le dispositif pour vous connecter à partir d'un terminal ASCII[1] ou pour vous connecter à l'appareil à partir d'un PC utilisant un logiciel d'émulation de terminal. Il y a une connexion RJ45 à une extrémité du câble et une connexion DB-9 série faux modem à l'autre extrémité. Le connecteur RJ45 se connecte au dispositif et le connecteur DB-9 série faux modem se connecte à votre terminal ASCII ou à votre ordinateur personnel. Utilisez le câble adaptateur USB/Série pour connecter le câble à votre ordinateur personnel.

Remarque : Pour la configuration initiale, vous pouvez utiliser le câble de connexion RJ45-série qui est livré avec le dispositif ou bien vous pouvez créer un câble sur la base des spécifications des brochages figurant dans le tableau suivant. N'utilisez pas un câble Ethernet pour connecter le port de la console série à un réseau Ethernet.

Tableau 1. Brochages du port série. Décrit les brochages du port série pour le connecteur de console.

RJ45		DB9	
Numéro de broche	**Signal**	**Numéro de broche**	**Signal**
1	RTS	8	CTS
2	DTR	6	DSR
3	TXD	2	RXD
4	GND	5	GND
5	GND	5	GND
6	RXD	3	TXD
7	DSR	4	DTR
8	CTS	7	RTS

Port USB

Le panneau avant du dispositif a une interface USB conforme aux périphériques USB 2.0. Ce connecteur USB n'est pas activé et ne fournit donc aucune connexion.

Voyants

Le panneau frontal du dispositif de type 7199 a trois voyants autonomes.

Voyant d'erreur

Le voyant d'erreur ambre est allumé si un événement critique est détecté.

Voyant d'emplacement

Le voyant d'emplacement bleu est allumé s'il est activé par le microprogramme. Vous pouvez contrôler si ce voyant est allumé à partir de la ligne de commande. Le voyant reste allumé jusqu'à sa désactivation. Utilisez la commande **`locate-led`** dans l'interface de ligne de commande :

- Pour activer, entrez la commande suivante :

```
locate-led on
```

- Pour désactiver, entrez la commande suivante :

```
locate-led off
```

Voyant d'alimentation

Le voyant d'alimentation est allumé lorsque le dispositif est connecté à une source d'alimentation électrique et que vous avez allumé le dispositif.

- Le voyant d'alimentation vert est allumé lorsque le dispositif est allumé et qu'il est en fonctionnement.
- Si le voyant n'est pas allumé, cela signifie que le dispositif a été éteint.

Bouton d'alimentation

Le bouton d'alimentation se trouve sur le panneau frontal du dispositif. Appuyez sur le bouton pour :

- Allumer le dispositif.
- Commencer un arrêt en douceur (si le dispositif est déjà allumé).

Le fait d'appuyer sur le bouton et de le maintenir enfoncé pendant 5 secondes effectue un arrêt matériel immédiat.

Remarque : Lorsque vous appuyez sur le bouton d'alimentation pour éteindre le dispositif, du courant électrique continue d'y arriver. Pour couper complètement l'électricité au dispositif, déconnectez tous les câbles d'alimentation.

Connecteurs réseau

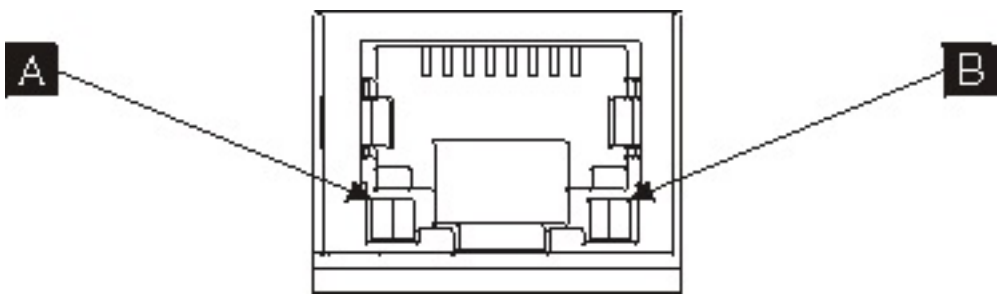
Le panneau frontal de chaque dispositif a deux ports Ethernet de gestion de réseau local et deux modules Ethernet. Voir Configuration du réseau Ethernet pour une description de la convention d'attribution de nom d'Ethernet.

Ports Ethernet de gestion de réseau local

Les deux ports Ethernet de gestion du système permettent la connexion au réseau local. Ces ports offrent un accès pour la gestion à distance de l'unité et ne peuvent pas être utilisés comme ports de données. Les interfaces Ethernet restantes peuvent gérer le trafic de données et les fonctions de connexion de et vers les différents services DataPower.

Méthode recommandée : Utilisez l'interface Ethernet `MGT0` ou `MGT1` pour les fonctions de gestion à l'échelle du système pour gérer le trafic réseau des fonctions entrantes SNMP, SSH et de l'interface utilisateur sur votre intranet.

Figure 3. Voyants des ports Ethernet



Connecteur Ethernet `MGT0`

Cette interface Ethernet peut gérer toutes les données de transactions sur le dispositif. Le connecteur Ethernet `MGT0` prend également en charge IPMI sur réseau local, y compris la liaison série sur réseau local. `MGT0` a un voyant de vitesse et un voyant d'activité associés :

Voyant de vitesse (A)

- Le voyant vert indique une connexion à 1 gigabits par seconde.

- Le voyant ambre indique une connexion à 10 ou à 100 mégabits par seconde.

Voyant d'activité (B)

- Le voyant vert indique que le port est lié.
- Le voyant vert clignotant indique que le port est actif.

Connecteur Ethernet MGT1

Cette interface Ethernet peut gérer toutes les données de transactions sur le dispositif. MGT1 a un voyant d'activité et un voyant de vitesse associés :

Voyant de vitesse (A)

- Le voyant vert indique une connexion à 1 gigabits par seconde.
- Le voyant ambre indique une connexion à 10 ou à 100 mégabits par seconde.

Voyant d'activité (B)

- Le voyant vert indique que le port est lié.
- Le voyant vert clignotant indique que le port est actif.

Modules Ethernet

Le dispositif DataPower a deux modules Ethernet pour la connectivité Ethernet. Le module Ethernet de gauche a huit ports RJ45 et le module Ethernet de droite a deux ports SFP+ (small-form factor pluggable) à 10 gigabits. Le nom de l'interface Ethernet dépend de la configuration du module, avec les noms des interfaces Ethernet dépendant de la configuration du module Ethernet.

Le module 1 gigabits prend en charge Ethernet avec du câble à paire torsadée non protégé, avec des interfaces standard, incluant :

- 10BASE-T
- 100BASE-TX
- 1000BASE-T

Le module 10 gigabits prend en charge les ports SFP+ (form-factor pluggable) avec des modules d'interface et des câbles de raccord. La négociation automatique est toujours activée :

10GBASE-SR
10GBASE-LR

Module Ethernet de gauche

A huit ports Ethernet à paire torsadée non protégée (RJ45). Les numéros Ethernet vont de ETH0 à ETH7 et correspondent au nombre de ports disponibles.

Module Ethernet de droite

A deux ports SFP à 10 gigabits. Les numéros Ethernet vont de ETH8 à ETH9 et correspondent au nombre de ports disponibles.

Voir Configuration du réseau Ethernet pour une description de la numérotation Ethernet.

Remarque : Les modules Ethernet ne sont pas remplaçables à chaud. Le remplacement à chaud des modules provoque une panne du système et peut éventuellement endommager votre dispositif.

Modules d'unité de disque dur

Le panneau avant du dispositif inclut quatre modules d'unité de disque dur de 2,5 pouces. Le dispositif prend en charge les unités de disque dur SAS et il y a deux voyants sur chaque module d'unité de disque dur. Le voyant gauche témoigne de l'activité du disque dur et le voyant droit indique un problème potentiel :

- Un voyant vert clignotant indique que des accès à l'unité de disque dur se produisent.
- Un voyant ambre clignotant indique que l'unité de disque dur est en erreur.
- Aucun voyant allumé indique que l'unité de disque dur n'est pas active.

Remarque : Les modules d'unité de disque dur ne sont pas remplaçables à chaud. Le remplacement à chaud des modules peut provoquer une panne du système.

Rubrique parent : Spécifications et fonctions du dispositif type 7199-92x

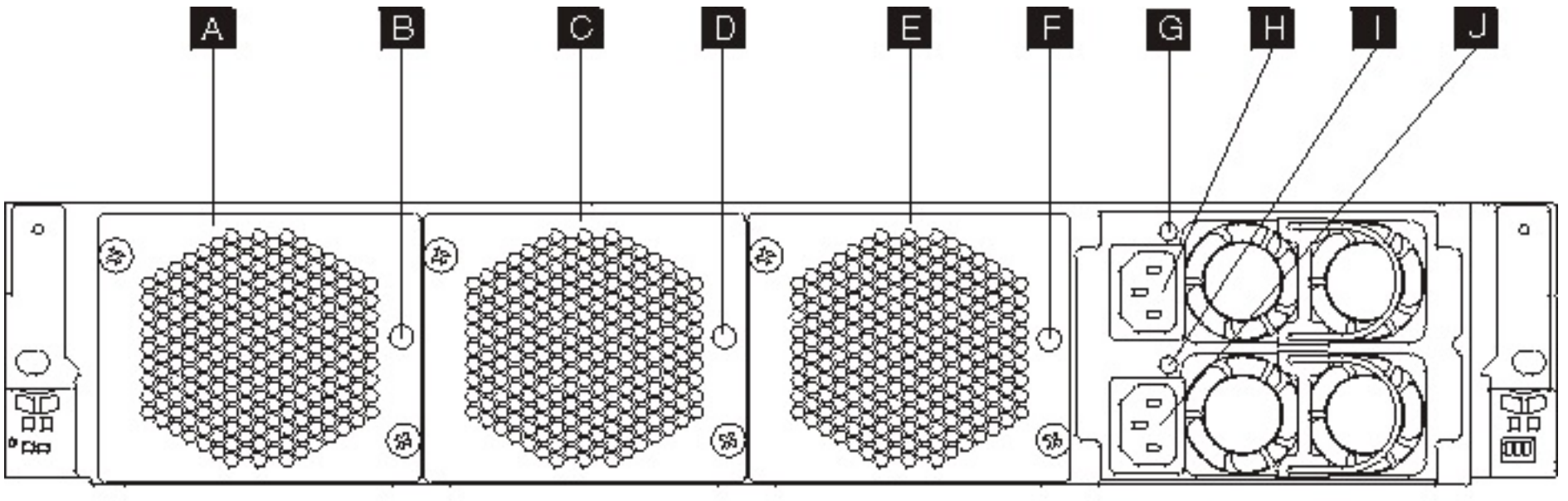
[1] Un périphérique simple qui transmet (entrées) et reçoit (sorties) des données ASCII.

Vue arrière du type 7199-92x

La vue arrière montre les composants et les voyants situés à l'arrière du dispositif. Les modules de ventilation et les modules d'alimentation sont installés depuis l'arrière du dispositif.

Diagramme de la vue arrière

Figure 1. Vue arrière du type 7199-92x



Les annotations du diagramme correspondent aux composants suivants du panneau arrière d'un dispositif de type 7199-92x :

A

Module de ventilation 1

B

Voyant du module de ventilation 1

C

Module de ventilation 2

D

Voyant du module de ventilation 2

E

Module de ventilation 3

F

Voyant du module de ventilation 3

G

Voyant du module d'alimentation 1

H

Module d'alimentation 1

E

Voyant du module d'alimentation 2

Identité J2EE (J

Module d'alimentation 2

Modules de ventilation

Le dispositif comprend trois modules de ventilation. Chaque module de ventilation contient un ventilateur de refroidissement individuel avec un voyant dans chaque module de ventilation :

- Si le voyant ambre est allumé, cela signifie qu'il y a un problème avec le module de ventilation.
- Si le voyant ambre n'est pas allumé, cela signifie que les ventilateurs fonctionnent normalement.

La vitesse des ventilateurs dépend de la température du dispositif. Quand la température augmente, la vitesse des ventilateurs augmente de façon à maintenir une température équilibrée pour le dispositif.

Modules d'alimentation

Le dispositif est alimenté par deux modules d'alimentation redondants. Un seul module d'alimentation suffit pour une prise en charge des opérations du dispositif. Les modules d'alimentation sont remplaçables à chaud : vous pouvez donc remplacer l'un de ceux-ci sans devoir mettre hors tension le dispositif. Chaque module d'alimentation comporte un voyant :

- Si le voyant d'alimentation ambre est allumé, c'est que l'alimentation a généré une erreur.
- Si le voyant ambre n'est pas allumé, cela signifie que le module d'alimentation fonctionne normalement.

Remarque : Lorsque vous appuyez sur le bouton d'alimentation pour éteindre le dispositif, le courant électrique continue d'y arriver. Pour couper complètement l'électricité au dispositif, déconnectez tous les câbles d'alimentation.

Rubrique parent : Spécifications et fonctions du dispositif type 7199-92x

Configuration du réseau Ethernet

Les modules Ethernet étendent les options de connectivité réseau. Chaque dispositif a deux modules Ethernet. Les modules Ethernet sont numérotés de gauche à droite, mais si un module a moins de huit ports, il utilisera le numéro de port le plus petit de la plage.

Convention de numérotation

La convention de numérotation pour la configuration des interfaces Ethernet et l'installation des câbles réseau est la suivante :

- Le module de gauche va de `ETH0` à `ETH7`
- Le module de droite va de `ETH8` à `ETH9`

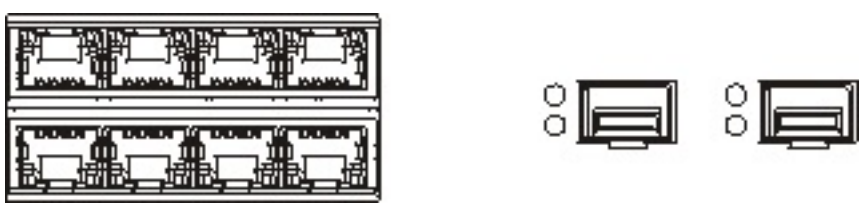
Connexions du type 7199

Chaque module Ethernet a une des configurations suivantes :

- Le module Ethernet de gauche a huit ports Ethernet à 1 gigabits, qui sont des connecteurs RJ45.
- Le module Ethernet de droite a deux ports Ethernet à 10 gigabits, qui sont des émetteurs-récepteurs SFP+ (small form-factor pluggable).

Le dispositif a dix connexions Ethernet. Les noms des interfaces Ethernet sont `ETH0` à `ETH7`, `ETH8` et `ETH9`.

Figure 1. Connexion Ethernet 8x2



Rubrique parent : Spécifications et fonctions du dispositif type 7199-92x

Interopérabilité du produit

WebSphere DataPower XC10 Appliance a été testé du point de vue de l'interopérabilité avec d'autres produits IBM®.

WebSphere Application Server

WebSphere Application Server doit implémenter des scénarios de session HTTP et de grille de données de mémoire cache dynamique. Pour plus d'informations sur les éditions spécifiques de WebSphere Application Server qui sont requises, voir Configuration requise.

WebSphere DataPower Integration Appliance XI50

Vous pouvez utiliser WebSphere DataPower XC10 Appliance en tant que cache secondaire avec WebSphere DataPower Integration Appliance XI50. La passerelle REST rend possible cette intégration en permettant aux clients non-Java d'accéder à des grilles de données simples avec un ensemble d'opérations HTTP. Pour plus d'informations sur la passerelle REST, voir Développement d'applications de grille de données avec la passerelle REST. Pour des informations et un exemple d'application montrant l'intégration entre WebSphere DataPower XC10 Appliance et WebSphere DataPower Integration Appliance XI50, voir Utilisation de WebSphere DataPower XC10 Appliance en tant que cache secondaire pour WebSphere DataPower XI50 Integration Appliance.

WebSphere Portal

Vous pouvez rendre persistantes des sessions HTTP à partir de WebSphere Portal dans une grille de données sur le dispositif. Pour plus d'informations sur la création de cette configuration, voir Configuration du gestionnaire de sessions HTTP avec WebSphere Portal. En outre, IBM Web Content Manager dans IBM WebSphere Portal peut utiliser des instances de mémoire cache dynamique pour stocker du contenu qui est extrait du gestionnaire de contenu Web lorsque la mise en cache avancée est activée. Le dispositif de mise en cache de WebSphere DataPower XC10 Appliance propose une implémentation de cache dynamique qui stocke le contenu mis en cache dans une grille de données élastique au lieu d'utiliser l'implémentation de mise en cache dynamique par défaut.

WebSphere Commerce

WebSphere Commerce Version 7.0.0.1 prend désormais en charge l'utilisation de WebSphere eXtreme Scale Client Version 7.1. Vous pouvez utiliser WebSphere DataPower XC10 Appliance pour mettre en mémoire cache les données du cache dynamique de WebSphere Commerce. Pour plus d'informations, voir Création de grilles de données en cache dynamique.

Tivoli Monitoring

Lorsque vous activez la surveillance SNMP pour WebSphere DataPower XC10 Appliance, vous pouvez importer ces données dans IBM Tivoli Monitoring version 6.2.2 groupe de correctifs 2 ou version ultérieure. Tivoli Monitoring permet de surveiller et gérer des applications système et réseau sur plusieurs systèmes d'exploitation, de contrôler la disponibilité et les performances de toutes les parties de votre entreprise et de générer des rapports pour le suivi des tendances et la résolution des incidents. Pour plus d'informations, voir Importing SNMP data from WebSphere DataPower XC10 Appliance into Tivoli Monitoring .

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Information associée:

Utilisation de WebSphere eXtreme Scale pour améliorer les performance WebSphere Portal et d'IBM Web Content Manager

.NET

Remarques relatives à Microsoft .NET

Deux environnements .NET existent dans WebSphere eXtreme Scale : l'environnement de développement et l'environnement d'exécution. Ces environnements ont des exigences spécifiques.

Configuration requise pour l'environnement de développement

Version de Microsoft .NET

La version .NET 3.5 et les versions suivantes sont prises en charge.

Microsoft Visual studio

Vous pouvez utiliser les versions suivantes de Visual Studio :

- Visual Studio 2008 SP1
- Visual Studio 2010 SP1
- Visual Studio 2012

Windows

Toute version de Windows compatible avec l'édition de Visual Studio que vous utilisez est compatible. Voir les liens suivants pour plus d'informations sur la configuration Windows requise pour Visual Studio :

- Configuration système requise pour Visual Studio 2008
- Configuration système requise pour Visual Studio 2010 Professional
- Configuration système requise pour Visual Studio 2012 Professional

Mémoire

- 1 Go (pour les installations 32 bits et 64 bits)

Espace disque

WebSphere eXtreme Scale nécessite 50 Mo d'espace disque disponible en plus des exigences Visual Studio.

Environnement d'exécution

Version de Microsoft .NET

La version .NET 3.5 et les versions suivantes sont compatibles, y compris l'exécution dans un environnement .NET 4.0 uniquement.

Windows

Tout environnement Windows qui remplit les exigences de version Microsoft .NET répertoriées plus haut.

Mémoire

65 Mo par processus qui accède aux données stockées sur les serveurs WebSphere eXtreme Scale.

Espace disque

WebSphere eXtreme Scale nécessite 35 Mo d'espace disque disponible. Lorsque la fonction de trace est activée, un espace disque supplémentaire de 2,5 Go est nécessaire.

Exécution d'WebSphere eXtreme Scale

Vous devez utiliser le mécanisme de transport eXtremeIO lorsque vous utilisez des applications client .NET. Pour plus d'informations sur eXtremeIO, voir Configuration d'IBM eXtremeIO (XIO).

2.5+ Configuration requise pour le fournisseur de stockage d'état de session ASP.NET

- Le fournisseur de stockage de session ASP.NET requiert l'une des versions de serveur IIS suivantes :
  - IIS 6.0 (fourni avec Windows Server 2003)
  - IIS 7.0 (fourni avec Windows Server 2008)
  - IIS 7.5 (fourni avec Windows Server 2008 R2)
  - IIS 8.0 (fourni avec Windows Server 2012)
- Mémoire : 120 Mo supplémentaires par processus (mémoire totale de 185 Mo).
- Sécurité : l'ID associé au pool d'applications ASP.NET doit disposer des droits d'administrateur.
- Sécurité : le niveau de confiance doit être Complet pour l'application ASP.NET.

Rubrique parent : Planification de l'environnement DataPower XC10 Appliance

Installation de WebSphere DataPower XC10 Appliance

Pour installer WebSphere DataPower XC10 Appliance au sein de votre environnement existant, vous devez dans un premier temps installer le matériel du dispositif. Installez ensuite WebSphere eXtreme Scale Client dans votre environnement d'application.

Démarrage rapide : Installation du matériel du dispositif
Avant de pouvoir commencer à utiliser le dispositif, vous devez l'installer dans l'armoire, configurer les accès nécessaires, démarrer l'interface utilisateur et vérifier l'état opérationnel du dispositif.

Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance
Une fois le dispositif DataPower XC10 Appliance physiquement installé et connecté, il est prêt à être initialisé et configuré.

Installation de WebSphere eXtreme Scale Client
Pour que IBM® WebSphere DataPower XC10 Appliance fonctionne avec WebSphere Application Server ou des clients dans un environnement autonome, vous devez installer WebSphere eXtreme Scale Client ou intégrer les fichiers JAR (Java™ ARchive) client à votre application.

Installation de profil Liberty
Vous installez l'environnement de service d'applications profil Liberty en utilisant le gestionnaire d'installation ou en exécutant un fichier d'archive Java (fichier JAR).

Identification et résolution des incidents liés à l'installation du produit
IBM Installation Manager est un programme d'installation commun à de nombreux logiciels IBM qui vous permet d'installer cette version de WebSphere eXtreme Scale.

Démarrage rapide : Installation du matériel du dispositif

Avant de pouvoir commencer à utiliser le dispositif, vous devez l'installer dans l'armoire, configurer les accès nécessaires, démarrer l'interface utilisateur et vérifier l'état opérationnel du dispositif.

Avant de commencer

- Vous devez disposer d'une connexion réseau à 1 gigabit pour le port de gestion `MGMT0`.
- Déterminez si vous utilisez les ports 1 gigabit ou 10 gigabits pour votre grille de données. Vous devez utiliser le module Ethernet de gauche (ports 1 gigabit) ou le module Ethernet de droite (ports 10 gigabits).
- Vous devez connaître les adresses IP des interfaces Ethernet pour l'accès à la gestion et l'utilisation de la grille de données.
- Vous devez connaître l'adresse IP des passerelles par défaut (routeurs) prenant en charge les sous-réseaux des interfaces Ethernet.
- Vous devez connaître l'adresse IP pour les services réseau (SSH, Telnet, etc.).

Procédure

1. Installez physiquement le dispositif dans l'armoire.

   Important : N'essayez pas d'ouvrir le boîtier du dispositif. L'ouverture du boîtier déclenche une anomalie de sécurité et le dispositif cesse alors de fonctionner. Dans ce cas, le dispositif devra être retourné à IBM® pour réinitialisation.

   a. Déballez le dispositif avec précaution. Localisez tous les cordons d'alimentation, câbles série et rails fournis.
   b. Identifiez l'emplacement d'installation du dispositif et assurez-vous que l'espace disponible au-dessus et au-dessous de celui-ci est suffisant pour sa ventilation et sa maintenance.
   c. Solidarisez les glissières de montage.
   d. Installez le dispositif sur les rails et faites-le glisser à sa place.
   e. En façade du dispositif, utilisez les câbles réseau pour connecter le dispositif à votre réseau. Les câbles Ethernet ne sont pas fournis.
      - Connectez le port de gestion `MGMT0` à un réseau à 1 gigabit.
      - Pour le trafic autre que le trafic de gestion pour la grille de données, utilisez le module Ethernet de gauche (ports 1 gigabit) ou le module Ethernet de droite (ports 10 gigabits).
   f. Utilisez les cordons d'alimentation fournis pour connecter les alimentations électriques aux prises de courant.
   g. A partir d'une console série, établissez une connexion avec le connecteur `CONSOLE` situé à l'avant du dispositif, en la configurant sur 9600 bauds 8N1 (8 bits par caractère, pas de parité, 1 bit d'arrêt) et sans contrôle de flux. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion. Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble. L'émulateur de terminal recommandé pour la console série est VT100.
   h. Appuyez sur le bouton d'alimentation. Le voyant d'alimentation vert s'allume et la console en série s'affiche.
   i. L'invite de connexion s'affiche. Pour la configuration initiale du dispositif, connectez-vous avec l'ID utilisateur et le mot de passe : xcadmin/xcadmin.

      Important : Ne perdez pas l'ID utilisateur et le mot de passe xcadmin. Si vous perdez ces informations, vous ne pouvez pas vous reconnecter au dispositif et vous devez le renvoyer à IBM pour réinitialisation ; toutes ses données seront alors effacées. Pour garantir l'accessibilité de l'ID utilisateur et du mot de passe xcadmin, vous pouvez configurer un serveur SMTP et une adresse électronique afin de permettre la réinitialisation du mot de passe xcadmin.

2. Configurez les accès au dispositif à l'aide de la console en série. Un assistant vous guide à travers le processus d'acceptation des contrats de licence et de configuration des ports Ethernet. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.
3. Mettez à jour le microprogramme de votre dispositif vers la version la plus récente. Pour plus d'informations, voir Mise à jour du microprogramme.
4. Suivez l'état de démarrage du dispositif après la mise à niveau du microprogramme. Vous pouvez suivre l'état de démarrage du dispositif dans l'interface utilisateur. Dans la barre d'adresse du navigateur Web, entrez : `https://<nom_hôte_dispositif>:9443/`. Vous pouvez également suivre la progression du démarrage en exécutant la commande **`start-progress`** dans le terminal que vous utilisez pour initialiser le dispositif. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.

5. Démarrez l'interface utilisateur.
    a. Dans la barre d'adresse du navigateur Web, entrez l'URL et le port définis lors de l'initialisation de l'unité. Vous pouvez utiliser l'adresse IP que vous avez définie ou le nom d'hôte qui correspond à l'adresse IP, par exemple `https://myXC10.ibm.com`. Utilisez le protocole HTTP sécurisé HTTPS.
    b. Entrez `xcadmin` dans la zone **Utilisateur** .
    c. Entrez le mot de passe correspondant dans la zone **Mot de passe**. Par défaut, ce mot de passe est `xcadmin`.
    d. Cliquez sur **Connexion**. Pour vous déconnecter, cliquez sur **Déconnexion**.
6. Vérifiez que le dispositif est opérationnel.
    - Le voyant vert sur le panneau frontal du dispositif est allumé.
    - Le voyant ambre d'erreur sur le panneau frontal du dispositif est éteint.
    - Le voyant vert de mise en cache sur le panneau frontal du dispositif doit s'allumer.
    - Le voyant ambre d'erreur de mise en cache sur le panneau frontal du dispositif est éteint.
    - L'afficheur LCD du panneau frontal du dispositif affiche le numéro de version du produit.

    Pour un diagramme montrant l'emplacement de ces voyants sur le dispositif, voir Spécifications et fonctions du dispositif .

    Si vous avez des problèmes, contactez le support IBM. Accédez au site Web : http://www-947.ibm.com/support/entry/portal/overview/software/websphere/websphere_datapower_xc10_appliance .

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Tâches associées:
Mise à jour du microprogramme
Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Une fois le dispositif DataPower XC10 Appliance physiquement installé et connecté, il est prêt à être initialisé et configuré.

Avant de commencer

Pour effectuer la configuration initiale, vous devez utiliser une connexion série. Cette connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif. Si vous utilisez un PC comme console série, vous devez utiliser un programme de communication série conçu pour un PC fonctionnant sous Windows ou Linux. Vous pouvez utiliser un matériel dédié, par exemple une console VT100, ou un PC exécutant un émulateur comme HyperTerminal ou Minicom. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion avec le dispositif.

Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de pouvoir continuer.

Avant de définir la configuration de base, rassemblez les informations suivantes :

- Assurez-vous que le port de gestion `MGMT0` dispose d'une connexion réseau d'un débit de 1 gigabit.
- Déterminez si vous utilisez les ports 1 gigabit ou 10 gigabits pour votre grille de données. Vous devez utiliser le module Ethernet de gauche (ports 1 gigabit) ou le module Ethernet de droite (ports 10 gigabits).
- Adresse IP et masque de sous-réseau de l'interface Ethernet pour l'accès à la gestion du dispositif (`MGMT`).
- Adresse IP des passerelles par défaut (routeurs) prenant en charge les sous-réseaux des interfaces Ethernet.
- Adresse IP du serveur DNS (Domain Name System).
- Informations sur le serveur de messagerie (pour la configuration des notifications par courrier électronique). Pour plus d'informations, voir Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance.
- Informations sur le serveur NTP (Network Time Protocol) pour permettre les communications entre les dispositifs de la collectivité.

Pourquoi et quand exécuter cette tâche

Utilisez cette tâche afin d'initialiser votre DataPower XC10 Appliance pour la première fois. La procédure à suivre pour effectuer votre première connexion au dispositif est légèrement différente de celle que vous utiliserez lors des connexions ultérieures.

ATTENTION :

1. **Consultez les informations importantes sur la protection de l'ID utilisateur et du mot de passe xcadmin qui se trouvent dans la rubrique Mot de passe xcadmin.**
2. **N'essayez pas d'ouvrir le boîtier du dispositif. L'ouverture du boîtier déclenche une anomalie de sécurité et le dispositif cesse alors de fonctionner. Dans ce cas, vous devez exécuter la commande `device clear-intrusion` pour rétablir le fonctionnement du dispositif.**
3. **Ne perdez pas le câble série fourni avec le dispositif. Vous en aurez besoin pour la configuration initiale du dispositif et peut-être ultérieurement pour l'identification des incidents. Ce câble a été spécialement conçu pour fonctionner avec le dispositif. Un autre câble ne fonctionnerait pas forcément avec le dispositif.**

Effectuez la configuration initiale de base du microprogramme. Il s'agit ici de la configuration minimale en vue d'ajouter un système WebSphere DataPower XC10 Appliance à votre environnement.

Procédure

1. Initialisez le dispositif. Procédez comme suit :
    a. Connectez la console série au dispositif à l'aide du câble série ou du câble USB/série fourni. Vous devez connecter le câble au connecteur CONSOLE situé sur la face avant du dispositif et le terminal ASCII ou le PC exécutant un logiciel d'émulation de terminal doit être opérationnel. Cette connexion vous permet voir les messages émis par le dispositif lors de son démarrage. Configurez le logiciel d'émulation avec 9600 baud 8N1 (8 bits par caractère, pas de parité, 1 bit d'arrêt) et sans contrôle de flux. L'émulateur de terminal recommandé pour la console série est VT100.
    b. Assurez-vous que le dispositif est sous tension. Si le dispositif est hors tension, appuyez sur le bouton d'alimentation. Le bouton d'alimentation se trouve sur le panneau frontal du dispositif.

Patientez quelques secondes pendant le démarrage du dispositif. Une fois le dispositif sous tension :

- Le voyant vert s'allume sur le panneau frontal du dispositif et le ventilateur se met en marche.
- L'invite de connexion s'affiche dans la console série. Pour la configuration initiale du dispositif, connectez-vous avec l'ID utilisateur et le mot de passe par défaut : xcadmin/xcadmin.

c. Acceptez les contrats de licence WebSphere DataPower XC10 Appliance. La première fois, avant de poursuivre, vous devez accepter les licences. Saisissez `Accept`, `Reject²` ou `StartOver` à chaque invite de licence.

d. Configurez le port Ethernet `MGMT`. Spécifiez l'adresse IP au format de routage interdomaine sans classes (CIDR).

e. Configurez la passerelle par défaut du port Ethernet `MGMT`.

f. Configurez les ports Ethernet pour votre grille de données. Spécifiez si vous utilisez des ports 1 gigabit ou des ports 10 gigabits. Configurez l'adresse CIDR pour les ports applicables.

g. Configurez les serveurs DNS (Domain Name System). Spécifiez une adresse IP valide pour votre serveur DNS.

2. Mettez à jour le microprogramme du dispositif. Pour télécharger ou mettre à jour le microprogramme de WebSphere DataPower XC10 Appliance, vous devez disposer des droits d'administration du dispositif. Le dispositif ne doit pas nécessairement être lui-même connecté à Internet pour extraire la mise à jour du microprogramme. Pour plus d'informations sur le téléchargement ou la mise à jour du microprogramme, reportez-vous à la rubrique Mise à jour du microprogramme.

3. Si vous installez le microprogramme pour la première fois sur un nouveau dispositif, vous devez exécuter la commande **`clear-all`** sur le dispositif. Effectuez les étapes suivantes après le redémarrage qui suit la mise à niveau du microprogramme :

   a. Etablissez une connexion avec le dispositif en tant qu'utilisateur `xcadmin`. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.

   b. Exécutez la commande **`clear-all`**.

   ```
   Console> clear-all
   Force Stopped all XC-10 processes
   Deleting configuration data and logs
   Deleting grid data
   ```

   ATTENTION :

   N'exécutez aucune autre commande avant la commande `clear-all`. L'exécution d'autres commandes peut créer des problèmes dans la configuration de votre dispositif.

   ATTENTION :

   N'exécutez aucune autre commande pendant que la commande `clear-all` est en cours d'exécution. Si vous voulez surveiller la progression du démarrage, n'utilisez que la commande `start-progress` ou la page correspondante de l'interface utilisateur du dispositif.

   c. Une fois que vous avez exécuté la commande **`clear-all`**, vous pouvez suivre la progression du démarrage du dispositif. Utilisez pour cela l'une des options suivantes :

   - 2.5+ Utilisez l'interface utilisateur du dispositif. Dans la barre d'adresse du navigateur Web, entrez l'URL et le port suivants : `https://<nom-hôte_dispositif>:9443/`. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.
   - Exécutez la commande **`start-progress`**. Lorsque cette commande renvoie `STARTED`, cela signifie que le dispositif est prêt à l'emploi.

4. Pour sécuriser la configuration, modifiez le mot de passe de l'utilisateur xcadmin. Le mot de passe par défaut est `xcadmin`. Vous pouvez modifier ce mot de passe à l'aide de la commande suivante :

   ```
   user password <ancien_mot_de_passe> <nouveau_mot_de_passe>
   ```

5. Vérifiez la configuration. Utilisez l'interface utilisateur avec un navigateur Web pour vérifier la configuration.

   Avertissement : La procédure de vérification ci-dessous suppose que le nom d'hôte de l'interface Ethernet est `myXC10.ibm.com`.

Pour accéder à l'interface utilisateur à partir d'un navigateur, procédez comme suit :

a. Ouvrez un navigateur Web. Ouvrez votre navigateur Web à partir d'un ordinateur connecté au réseau.
b. Entrez l'URL. Dans la barre d'adresse, entrez l'URL défini lors de l'initialisation du dispositif. Par exemple : `https://myXC10.ibm.com`.

   Remarque : Utilisez le protocole `https` et non pas `http`.

c. Connectez-vous au dispositif. Connectez-vous au dispositif à l'aide du compte `XCADMIN` local et du mot de passe. Le mot de passe que vous tapez est en clair ; pour des raisons de sécurité il ne s'affiche donc pas.
d. Cliquez sur **Connexion**.

   Si la page Bienvenue s'affiche, l'authentification du compte `XCADMIN` local a abouti.

6. Effectuez la configuration.

Résultats

La configuration initiale pour WebSphere DataPower XC10 Appliance est terminée.

Que faire ensuite

Configurez les paramètres de votre dispositif dans l'interface utilisateur. Ces paramètres incluent les utilisateurs et groupes d'utilisateurs, la sécurité, la distribution par courrier électronique, les serveurs de noms de domaine (DNS) et les paramètres de date et d'heure. Pour plus d'informations, voir Configuration de votre dispositif.

Mot de passe xcadmin
Après avoir défini l'ID et le mot de passe de l'administrateur (xcadmin), conservez-les dans un endroit sûr. Vous pouvez configurer la distribution par courrier électronique de sorte à permettre la réinitialisation des mots de passe dans l'interface utilisateur.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Tâches associées:
Mise à jour du microprogramme
Démarrage rapide : Installation du matériel du dispositif

Mot de passe xcadmin

Après avoir défini l'ID et le mot de passe de l'administrateur (xcadmin), conservez-les dans un endroit sûr. Vous pouvez configurer la distribution par courrier électronique de sorte à permettre la réinitialisation des mots de passe dans l'interface utilisateur.

Sauvegarde de l'ID utilisateur et du mot de passe

L'ID utilisateur et le mot de passe xcadmin étant nécessaires pour se connecter au dispositif, placez-les en lieu sûr une fois que vous les avez modifiés. Si vous les perdez et que vous n'avez pas moyen de les récupérer, vous devez renvoyer le dispositif à IBM® en vue de sa réinitialisation

Réinitialisation des mots de passe par courrier électronique

Si vous configurez la distribution par courrier électronique, tous les utilisateurs peuvent restaurer leur mot de passe en cliquant sur le lien **Mot de passe oublié ?** dans l'écran de connexion de l'interface utilisateur. Un courrier électronique contenant un nouveau mot de passe généré est envoyé à l'utilisateur.

ATTENTION :

La seule manière de réinitialiser le mot de passe xcadmin est d'utiliser le lien Mot de passe oublié ? dans l'écran de connexion de l'interface utilisateur. Si vous oubliez le mot de passe et que la distribution par courrier électronique n'est pas configurée, vous devez réinitialiser le dispositif avec la commande `device RESET`, ce qui réinitialise tous les paramètres du dispositif. Voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif pour plus d'informations sur la commande `device RESET`.

Modification du mot de passe xcadmin

Vous pouvez éditer le mot de passe de xcadmin dans l'interface utilisateur ou l'interface de ligne de commande du dispositif.

Pour modifier le mot de passe dans l'interface utilisateur, éditez l'utilisateur. Pour plus d'informations, voir Gestion des utilisateurs.

Pour modifier le mot de passe dans l'interface de ligne de commande du dispositif, vous pouvez utiliser la commande **`user password`**. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Gestion du serveur DNS (Domain Name System)
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance

Référence associée:
Passerelle REST : Configuration de la sécurité

Installation de WebSphere eXtreme Scale Client

Pour que IBM® WebSphere DataPower XC10 Appliance fonctionne avec WebSphere Application Server ou des clients dans un environnement autonome, vous devez installer WebSphere eXtreme Scale Client ou intégrer les fichiers JAR (Java™ ARchive) client à votre application.

Pourquoi et quand exécuter cette tâche

L'installation de WebSphere eXtreme Scale Client est requise pour que vos applications client communiquent avec IBM WebSphere DataPower XC10 Appliance. Avant d'installer le client, vous devez déterminer si vous souhaitez l'installer dans un environnement autonome ou dans un environnement WebSphere Application Server.

Si vos applications utilisent les sessions HTTP ou la mémoire cache dynamique de WebSphere Application Server, vous devez installer WebSphere eXtreme Scale Client dans l'environnement WebSphere Application Server.

Des grilles de données simples peuvent être utilisées dans un environnement WebSphere Application Server imbriqué ou dans un environnement autonome. Lors de l'installation du client dans un environnement autonome, l'installation n'utilise pas WebSphere Application Server.

Pour plus d'informations sur les types de grille de données, voir Topologie du dispositif : collectivités, zones et grilles de données.

Procédure

1. Téléchargez et installez IBM Installation Manager et vérifiez que vous avez installé les référentiels de produit nécessaires. Voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.
2. Téléchargez WebSphere eXtreme Scale Client à partir du site de support. Pour plus d'informations sur l'emplacement de téléchargement du client, voir le Portail de support.
3. Exécutez l'installation à l'aide d'Installation Manager. Choisissez l'offre de produit correcte. Les offres de produit sont disponibles selon les référentiels que vous avez ajoutés à vos préférences d'installation dans Installation Manager. Les offres de produit disponibles pour WebSphere eXtreme Scale Client sont les suivantes :
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 6
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

   Utilisez l'installation qui convient à votre configuration planifiée :
   - **Installation client imbriquée** : Si vous utilisez la mémoire cache dynamique ou les sessions HTTP, vous devez utiliser l'installation imbriquée. Vous pouvez également utiliser l'installation imbriquée avec des grilles de données simples. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
   - **Installation client autonome** : Vous ne pouvez utiliser l'installation client que dans un environnement autonome comportant des grilles de données simples. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.
   - **Installation en mode silencieux** : Vous pouvez également installer le client dans un environnement autonome ou dans un environnement WebSphere Application Server imbriqué, à l'aide d'un fichier de réponses. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses.

Que faire ensuite

Configurez le dispositif. Pour plus d'informations, voir Configuration de votre dispositif.

Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client
Les offres de produit WebSphere eXtreme Scale Client sont disponibles dans les référentiels du produit. Avant de pouvoir accéder à ces référentiels, vous devez installer IBM Installation Manager.

Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager
Utilisez IBM Installation Manager pour désinstaller des offres de produit WebSphere eXtreme Scale Client.

.NET 2.5+ **Présentation de l'installation de WebSphere eXtreme Scale Client for .NET**
Vous pouvez installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution ou dans un environnement d'exécution et de production.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Java

Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Les offres de produit WebSphere eXtreme Scale Client sont disponibles dans les référentiels du produit. Avant de pouvoir accéder à ces référentiels, vous devez installer IBM® Installation Manager.

Vous pouvez installer Installation Manager en utilisant les fichiers disponibles sur le support du produit, ou en utilisant un fichier obtenu à partir du site Passport Advantage ou en utilisant un fichier disponible sur le site Web de téléchargement d'IBM Installation Manager. Un fichier est un fichier compressé contenant des images d'installation.

Remarque :

Installation Manager est disponible pour le téléchargement en version 32 et 64 bits. Vous pouvez utiliser une version d'Installation Manager pour installer WebSphere eXtreme Scale.

Installation Manager vous permet d'accéder aux référentiels de produit nécessaires. Vous devez accéder à ces référentiels afin d'installer les offres de produit WebSphere eXtreme Scale.

Vous disposez de deux options pour accéder aux référentiels de produit.

Option 1 : Accédez aux référentiels de produit sur le support physique et utilisez une installation locale

1. Installez Installation Manager sur votre système.
2. Utilisez Installation Manager pour installer l'offre de produit à partir des référentiels du produit présents sur le support.

Option 2 : Téléchargez les référentiels de produit à partir de Passport Advantage et utilisez une installation locale

1. Téléchargez les référentiels à partir du site Passport Advantage.

   Remarque : Voir Voir Logiciels pris en charge pour obtenir la liste des images d'installation IBM WebSphere eXtreme Scale téléchargeables depuis le site Web IBM Passport Advantage et d'autres informations.
2. Installez Installation Manager sur votre système.
3. Utilisez Installation Manager pour installer le produit à partir des référentiels de produit téléchargés.

ID d'offre pour les produits WebSphere eXtreme Scale Client
Lors de l'installation des mises à jour de produit ou lors de l'annulation de correctifs, vous êtes tenu d'indiquer l'ID d'offre à partir de la ligne de commande. Utilisez le tableau ci-dessous pour identifier l'offre de produit.

Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server
Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement dans lequel WebSphere Application Server ou WebSphere Application Server Network Deployment est installé. Vous pouvez utiliser les fonctions existantes de WebSphere Application Server ou WebSphere Application Server Network Deployment pour améliorer vos applications WebSphere DataPower XC10 Appliance.

Installation d'IBM Installation Manager à l'aide de l'interface graphique
Pour accéder aux référentiels de produit nécessaires à l'installation des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM Installation Manager. Vous pouvez installer Installation Manager à l'aide d'une console d'assistant.

Installation d'IBM Installation Manager à l'aide de la ligne de commande
Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM Installation Manager. Vous pouvez installer Installation Manager à partir de la ligne de commande.

Installation d'IBM Installation Manager à l'aide de fichiers de réponses
Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM Installation Manager. Vous pouvez installer Installation Manager à l'aide de fichiers de réponses.

Rubrique parent : Installation de WebSphere eXtreme Scale Client

ID d'offre pour les produits WebSphere eXtreme Scale Client

Lors de l'installation des mises à jour de produit ou lors de l'annulation de correctifs, vous êtes tenu d'indiquer l'ID d'offre à partir de la ligne de commande. Utilisez le tableau ci-dessous pour identifier l'offre de produit.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server

Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement dans lequel WebSphere Application Server ou WebSphere Application Server Network Deployment est installé. Vous pouvez utiliser les fonctions existantes de WebSphere Application Server ou WebSphere Application Server Network Deployment pour améliorer vos applications WebSphere DataPower XC10 Appliance.

Avant de commencer

- Vérifiez que le répertoire d'installation cible ne contient pas une installation de WebSphere eXtreme Scale Client.
- Arrêtez tous les processus en cours d'exécution dans votre environnement WebSphere Application Server ou WebSphere Application Server Network Deployment. Voir Utilitaires de ligne de commande pour plus d'informations sur les commandes **stopManager**, **stopNode** et **stopServer**.

  ATTENTION :

  Vérifiez que les processus actifs sont arrêtés. Si les processus en cours d'exécution ne sont pas arrêtés, l'installation se poursuit avec des résultats imprévisibles. Sur certaines plateformes, l'installation peut prendre fin dans un état indéterminé.

Important : Lorsque vous installez WebSphere eXtreme Scale Client, ce doit être dans le répertoire dans lequel vous avez installé WebSphere Application Server. Par exemple, si vous avez installé WebSphere Application Server dans `C:\racine_was`, vous devez également choisir `C:racine_was` comme répertoire cible lors de l'installation de WebSphere eXtreme Scale Client.

Pourquoi et quand exécuter cette tâche

Intégrez eXtreme Scale à WebSphere Application Server ou WebSphere Application Server Network Deployment pour appliquer les fonctions d'eXtreme Scale à vos applications Java™ Platform, Enterprise Edition. Les applications Java EE y accèdent à l'aide d'une connexion client.

Procédure

- Pour installer WebSphere eXtreme Scale Client dans un environnement WebSphere Application Server version 8, procédez comme suit :
  1. Installez IBM Installation Manager. Pour plus d'informations, voir Installation d'IBM Installation Manager à l'aide de l'interface graphique.
  2. A l'aide d'Installation Manager, installez l'offre de produit eXtreme Scale appropriée :
     - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

     Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.
  3. Téléchargez les référentiels WebSphere Application Server version 8 nécessaires à partir du site Passport Advantage. Pour plus d'informations, voir How to download WebSphere Application Server - Express V8.5 from Passport Advantage.
  4. Installez WebSphere Application Server version 8. Pour plus d'informations, voir Installation du produit sur des systèmes d'exploitation distribués à l'aide de l'interface graphique.
- Pour installer WebSphere eXtreme Scale Client dans un environnement WebSphere Application Server version 7, procédez comme suit :
  1. Installez IBM Installation Manager. Pour plus d'informations, voir Installation d'IBM Installation Manager à l'aide de l'interface graphique.
  2. Installez WebSphere Application Server Version 7 en utilisant le programme d'installation InstallShield MultiPlatform (ISMP). Pour plus d'informations, voir Installation de l'environnement de service d'applications.
  3. Après l'installation, vous devez importer WebSphere Application Server version 7 dans le gestionnaire d'installation pour terminer l'installation. En important WebSphere Application Server version 7 dans Installation Manager, vous pouvez gérer et installer des groupes de correctifs pour le produit à partir d'un seul emplacement. Vous devez veiller à ce que les référentiels nécessaires soient configurés dans Installation Manager pour l'accès aux groupes de correctifs et aux mises à jour. Pour plus d'informations sur l'importation d'une installation existante de WebSphere Application Server 7 dans Installation Manager, voir Importation des infos produit d'IBM WebSphere Application Server dans le registre d'IBM Installation Manager.
  4. A l'aide d'Installation Manager, installez l'offre de produit eXtreme Scale appropriée :

- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7

Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation d'IBM Installation Manager à l'aide de l'interface graphique

Pour accéder aux référentiels de produit nécessaires à l'installation des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM® Installation Manager. Vous pouvez installer Installation Manager à l'aide d'une console d'assistant.

Avant de commencer

Vous devez installer IBM Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Accédez à l'emplacement contenant les fichiers d'installation du gestionnaire d'installation et exécutez l'une des commandes suivantes :

   Installation par un administrateur :

   - Windows `install.exe`
   - UNIX Linux `./install`

   Installation par un non administrateur :

   - Windows `userinst.exe`
   - UNIX Linux `./userinst`

   Pour plus d'informations sur les installations administratives et non administratives, voir Installation en tant qu'administrateur, non-administrateur ou groupe

   Installation en mode groupe :

   UNIX Linux `./groupinst`

   Remarques sur le mode groupe :

   - Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des progiciels.

     Le mode Groupe ne signifie pas que deux personnes peuvent utiliser la même instance d'IBM Installation Manager en même temps.
   - Windows Le mode groupe n'est pas disponible sur les systèmes d'exploitation Windows.
   - Si vous n'installez pas le gestionnaire d'installation avec le mode Groupe, vous ne pourrez pas utiliser le mode Groupe pour gérer les produits que vous installez ultérieurement à l'aide de cette instance d'Installation Manager.
   - Remplacez l'emplacement d'installation par défaut de l'utilisateur en cours par un emplacement accessible à tous les utilisateurs du groupe.
   - Configurez vos groupes, droits d'accès et variables d'environnement, comme décrit dans les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5 avant d'effectuer l'installation en mode groupe.
   - Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

2. Veillez à sélectionner le package du gestionnaire d'installation et cliquez sur **Next**.
3. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
4. Cliquez sur **Suivant**.
5. Lisez le récapitulatif et cliquez sur **Installer**. Si l'installation a réussi, le programme affichera un message pour confirmer le succès de l'installation. Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
6. Ajoutez le référentiel du produit à vos préférences Installation Manager.
   a. Démarrez Installation Manager.
   b. Dans le menu du haut, cliquez sur **Fichier > Préférences**.
   c. Sélectionnez **Repositories**.
   d. Cliquez sur **Add Repository**.
   e. Entrez le chemin du fichier `repository.config` dans l'emplacement contenant les fichiers du référentiel :

- Windows `C:\repositories\nom_produit\local-repositories`
- UNIX Linux `/var/repositories/nom_produit/local-repositories`

f. Cliquez sur **OK**.

7. Effacez tous les emplacements énumérés dans la fenêtre Repositories, qui ne seront pas utilisés.
8. Cliquez sur **Apply**.
9. Cliquez sur **OK**.
10. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Que faire ensuite

Lorsque l'installation du gestionnaire d'installation et la configuration du référentiel aboutissent, vous pouvez continuer d'installer WebSphere eXtreme Scale Client ou for pour l'offre de produit. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique
Utilisez Installation Manager à partir de la console de l'assistant pour installer les offres de produit WebSphere eXtreme Scale Client.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Utilisez Installation Manager à partir de la console de l'assistant pour installer les offres de produit WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Démarrez Installation Manager.

   UNIX Linux **Conseil :** Vous pouvez démarrer Installation Manager en mode groupe avec la commande **./IBMIM**.

   - Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des packages logiciels.
   - Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

2. Cliquez sur **Install**.

   Remarque : Si vous êtes invité à vous authentifier, utilisez l'ID et le mot de passe IBM avec lesquels vous vous êtes enregistré sur le site Web du programme.

   Installation Manager analyse ses référentiels définis à la recherche des packages disponibles.

3. Sélectionnez l'une des offres de produit ci-dessous ainsi que la version appropriée.
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

   Si le produit est déjà installé sur votre système, un message s'affichera pour vous en informer. Pour créer une autre installation du produit dans un autre emplacement, cliquez sur **Continuer**.

   Conseil : Si l'option **Search service repositories during installation and updates** est sélectionnée sur la page des préférences de référentiel dans Installation Manager et que vous êtes connecté à Internet, vous pouvez cliquer sur **Check for Other Versions and Extensions**. Ainsi, vous pouvez rechercher les mises à jour dans les référentiels de mise à jour par défaut pour les packages sélectionnés. Dans ce cas, il n'est pas nécessaire d'ajouter l'URL de référentiel de service spécifique à la page des préférences en matière de référentiel d'Installation Manager.

   a. Sélectionnez les correctifs à installer.

      Les correctifs recommandés sont sélectionnés par défaut.

      Vous pouvez choisir de n'afficher que les correctifs recommandés et de masquer les autres.

   b. Cliquez sur **Suivant**.

      Remarque : Installation Manager peut vous inviter à effectuer une mise à jour vers le dernier niveau d'Installation Manager lors de la connexion au référentiel. Si vous y êtes invité, effectuez la mise à jour vers la nouvelle version avant de poursuivre. Pour plus d'informations sur les mises à jour automatiques, voir le centre de documentation d'IBM Installation Manager Version 1.5.

4. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
5. Indiquez le répertoire principal d'installation du produit.

   Le panneau affiche également le répertoire de ressources partagées et les informations d'espace disque.

   Remarque : La première fois que vous installez un package à l'aide d'Installation Manager, spécifiez le répertoire des ressources partagées. Le répertoire des ressources partagées est l'endroit où les artefacts d'installation se trouvent ; ils peuvent être utilisés par un plusieurs groupe de packages. Utilisez votre unité ayant la taille la plus importante pour cette installation. Vous ne pouvez pas

changer l'emplacement du répertoire tant que vous n'avez pas désinstallé tous les packages.

Restrictions :

- Si vous supprimez l'emplacement cible par défaut et que vous ne renseignez pas une zone du répertoire d'installation, vous ne pouvez pas continuer.
- N'utilisez pas de liens symboliques comme répertoire de destination.

  car ils ne sont pas pris en charge.
- N'utilisez pas de point-virgule dans le nom d'un répertoire.

  Si le répertoire de destination inclut un point-virgule, cela signifie que WebSphere eXtreme Scale n'est pas installé comme prévu.

  Windows Sous Windows, le point-virgule est le caractère utilisé pour construire le chemin d'accès aux classes.

  - Windows La longueur maximale du chemin d'accès sur les systèmes d'exploitation Windows Server 2008, Windows Vista et Windows 7 est de 60 caractères.

6. Cliquez sur **Suivant**.
7. Sélectionnez les langues pour lesquelles un contenu traduit doit être installé.

   L'anglais est toujours sélectionné.
8. Cliquez sur **Suivant**.
9. Sélectionnez les fonctions que vous voulez installer.

   En fonction de l'offre de produit sélectionnée, vous pouvez choisir l'une des fonctions suivantes :
   - Console

     Disponible pour toutes les offres de produit WebSphere eXtreme Scale. Vous pouvez choisir d'installer la console de surveillance. Avec la console Web, vous pouvez générer des graphiques des statistiques actuelles et historiques. Cette console fournit un certain nombre de graphiques pour des présentations générales et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.
   - Samples

     Disponible pour toutes les offres de produit WebSphere eXtreme Scale.
10. Cliquez sur **Suivant**.
11. Lisez le récapitulatif et cliquez sur **Installer**.

    - Si l'installation aboutit, le programme affiche un message indiquant que l'installation a abouti.

      Remarque : Le programme peut également afficher d'importantes instructions post-installation.
    - Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
12. Sélectionnez l'outil que vous souhaitez démarrer à la fin de l'installation.
    - Sélectionnez **Outil de gestion des profils pour créer un profil** si vous souhaitez créer un profil de serveur d'applications avec des paramètres appropriés pour un environnement de production.
    - Sélectionnez **Outil de gestion des profils pour créer un profil de serveur d'applications pour un environnement de développement** si vous souhaitez créer un profil de serveur d'applications avec des paramètres appropriés pour un environnement de développement.

      Remarque : Les paramètres de développement sont adaptés à un environnement de développement au sein desquelles des mises à jour sont effectuées et les ressources systèmes sont au minimum. N'utilisez pas les paramètres de développement pour les serveurs de production.
    - Sélectionnez **Aucun** si vous ne souhaitez pas créer de profil lorsque l'installation est terminée.

    Restriction : L'option de démarrage de l'outil de gestion des profils est disponible uniquement si une version de WebSphere Application Server contenant l'outil de gestion des profils est installée.
13. Cliquez sur **Terminer**.

14. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Rubrique parent : Installation d'IBM Installation Manager à l'aide de l'interface graphique

Tâches associées:
Désinstallation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Java

Installation d'IBM Installation Manager à l'aide de la ligne de commande

Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM® Installation Manager. Vous pouvez installer Installation Manager à partir de la ligne de commande.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Accédez à l'emplacement contenant les fichiers d'installation d'Installation Manager et exécutez une des commandes suivantes :

   Installation par un administrateur :

   - Windows `installc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
   - UNIX Linux `./installc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

   Installation par un non administrateur :

   - Windows `userinstc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
   - UNIX Linux `./userinstc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

   Installation en mode groupe :

   UNIX Linux `./groupinstc -acceptLicense -dataLocation` *`emplacement_données_application`* `-log` *`chemin_et_nom_fichier_journal`* `-installationDirectory` *`rép_Installation_Manager`*

   Remarques sur le mode groupe :

   - Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des packages logiciels.
   - Windows Le mode groupe n'est pas disponible sur les systèmes d'exploitation Windows.
   - Si vous n'installez pas Installation Manager avec le mode groupe, vous ne pourrez pas utiliser le mode groupe pour gérer les produits que vous installez ultérieurement à l'aide de cette instance d'Installation Manager.
   - Assurez-vous de bien changer l'emplacement d'installation et de le faire passer de l'emplacement par défaut dans le répertoire de base de l'utilisateur actuel à un emplacement accessible par tous les utilisateurs du groupe.
   - Configurez vos groupes, droits d'accès et variables d'environnement, comme décrit dans les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5 avant d'effectuer l'installation en mode groupe.
   - Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

2. Facultatif : Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

   Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

   Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

Que faire ensuite

Une fois que l'installation d'Installation Manager et la configuration du référentiel aboutissent, vous pouvez continuer d'installer l'offre de produit WebSphere eXtreme Scale WebSphere eXtreme Scale Client ou WebSphere eXtreme Scale for WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.

Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande
Utilisez Installation Manager à partir de la ligne de commande pour installer les offres de produit WebSphere eXtreme Scale Client.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Utilisez Installation Manager à partir de la ligne de commande pour installer les offres de produit WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

1. Ouvrez une session sur votre système.
2. Accédez au sous-répertoire `eclipse/tools` dans le répertoire où vous avez installé Installation Manager.
3. Vérifiez que le référentiel du produit est disponible.

   Windows

   ```
   imcl.exe listAvailablePackages -repositories référentiel_source
   ```

   UNIX Linux

   ```
   ./imcl listAvailablePackages -repositories référentiel_source
   ```

   Vous devez voir un ou plusieurs niveaux de l'offre.
4. Utilisez la commande **imcl** pour installer le produit.

   Windows

   ```
   imcl.exe install com.ibm.websphere.v85_version_offre,ID_fonction_facultative
     -repositories référentiel_source
     -installationDirectory répertoire_installation
     -sharedResourcesDirectory répertoire_partagé
     -accessRights mode_accès
     -preferences clé_préférence=valeur
     -properties clé_propriété=valeur
     -keyring fichier_de_clés -password mot_de_passe
     -acceptLicense
   ```

   UNIX Linux

   ```
   ./imcl install com.ibm.websphere.version_offre,ID_fonction_facultative
     -repositories référentiel_source
     -installationDirectory répertoire_installation
     -sharedResourcesDirectory répertoire_partagé
     -accessRights mode_accès
     -preferences clé_préférence=valeur
     -properties clé_propriété=valeur
     -keyring fichier_de_clés -password mot_de_passe
     -acceptLicense
   ```

Conseils :

- *ID_offre* correspond à l'ID offre répertorié dans la rubrique ID d'offre pour les produits WebSphere eXtreme Scale Client.
- La *version_offre*, qui peut, le cas échéant, être associée à l'ID offre à l'aide d'un trait de soulignement, est une version spécifique de l'offre à installer (`8.6.0.20110503_0200`, par exemple).

  - Si la *version_offre* n'est **pas** spécifiée, la version la plus récente de l'offre et **tous** les correctifs temporaires de cette version sont installés.
  - Si la *version_offre* est spécifiée, la version indiquée de l'offre est installée et **aucun** correctif temporaire de cette version n'est installé.

  La version de l'offre peut être rattachée à la fin de l'ID offre à l'aide d'un trait de soulignement

lorsque vous exécutez la commande suivante au niveau du référentiel :

```
imcl listAvailablePackages -repositories référentiel_source
```

- Vous pouvez également spécifier `none`, `recommended` ou `all` avec l'argument `-installFixes` afin d'indiquer les correctifs temporaires que vous souhaitez installer avec l'offre.
  - Si la version de l'offre n'est **pas** spécifiée, l'option `-installFixes` est paramétrée par défaut sur `all`.
  - Si la version de l'offre est spécifiée, l'option `-installFixes` est paramétrée par défaut sur `none`.
- Vous pouvez ajouter une liste de fonctions séparées par des virgules. Exemple de programme :

```
imcl -acceptLicense install
com.ibm.websphere.WXS.v85,xs.console.feature,xs.samples.feature
```

```
imcl -acceptLicense install
com.ibm.websphere.WXS.v86,xs.console.feature,xs.samples.feature
```

  - `xs.console.feature` Disponible pour toutes les offres de produit. Vous pouvez choisir d'installer la console de surveillance. Avec la console Web, vous pouvez générer des graphiques des statistiques actuelles et historiques. Cette console fournit un certain nombre de graphiques préconfigurés pour des présentations générales et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.
  - `xs.samples.feature` Disponible pour toutes les offres de produit. Vous pouvez choisir d'installer des exemples.

Remarques :

- Si vous avez précédemment spécifié le mode d'installation d'Installation Manager, le paramètre -accessRights n'est pas requis.
- Si vous rencontrez des problèmes ultérieurement, Installation Manager peut sauvegarder les versions antérieures d'un package à des fins de rétrogradation. Quand Installation Manager rétrograde un package à une version antérieure, la version actuelle des fichiers est désinstallée et les versions antérieures sont réinstallées. Si vous choisissez de ne pas sauvegarder les fichiers pour rétrogradation, vous pouvez empêcher la sauvegarde des fichiers en indiquant une préférence :

  ```
  -preference
  com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts=False
  ```

  Pour plus d'informations sur la définition de vos préférences pour Installation Manager, voir le centre de documentation d'IBM Installation Manager Version 1.5.

  Conseil : Même si vous choisissez de ne pas sauvegarder les fichiers pour rétrogradation, vous pouvez accéder aux fichiers de produit pour rétrogradation à partir du référentiel.

- Le programme peut générer des instructions post-installation importantes dans la sortie standard.

  Pour plus d'informations sur l'utilisation de la commande **`imcl`** pour installer le produit, voir le centre de documentation d'IBM Installation Manager Version 1.5.

Rubrique parent : Installation d'IBM Installation Manager à l'aide de la ligne de commande

Tâches associées:
Désinstallation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Java

Installation d'IBM Installation Manager à l'aide de fichiers de réponses

Pour accéder aux référentiels de produit nécessaires afin d'installer des offres de produit WebSphere eXtreme Scale Client, vous devez installer IBM® Installation Manager. Vous pouvez installer Installation Manager à l'aide de fichiers de réponses.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Procédure

Accédez à l'emplacement contenant les fichiers d'installation d'Installation Manager, puis exécutez une des commandes ci-dessous pour installer Installation Manager.

Installation par un administrateur :

- Windows `installc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
- UNIX Linux `./installc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

Installation par un non administrateur :

- Windows `userinstc.exe -acceptLicense -log` *`chemin_et_nom_fichier_journal`*
- UNIX Linux `./userinstc -acceptLicense -log` *`chemin_et_nom_fichier_journal`*

Installation en mode groupe :

UNIX Linux `./groupinstc -acceptLicense -dataLocation` *`emplacement_données_application`* `-log` *`chemin_et_nom_fichier_journal`* `-installationDirectory` *`rép_Installation_Manager`*

Remarques sur le mode groupe :

- Le mode groupe permet à plusieurs utilisateurs d'utiliser une même instance d'IBM Installation Manager pour gérer des packages logiciels.

  Le mode groupe ne signifie pas que deux personnes peuvent utiliser la même instance d'IBM Installation Manager en même temps.

- Windows Le mode groupe n'est pas disponible sur les systèmes d'exploitation Windows.
- Si vous n'installez pas Installation Manager avec le mode groupe, vous ne pourrez pas utiliser le mode groupe pour gérer les produits que vous installez ultérieurement à l'aide de cette instance d'Installation Manager.
- Assurez-vous de bien changer l'emplacement d'installation et de le faire passer de l'emplacement par défaut dans le répertoire de base de l'utilisateur actuel à un emplacement accessible par tous les utilisateurs du groupe.
- Configurez vos groupes, droits d'accès et variables d'environnement, comme décrit dans les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5 avant d'effectuer l'installation en mode groupe.
- Pour plus d'informations sur l'utilisation du mode groupe, consultez les feuilles de route du mode groupe du centre de documentation d'IBM Installation Manager version 1.5.

Que faire ensuite

Une fois que l'installation d'Installation Manager et la configuration du référentiel aboutissent, vous pouvez continuer d'installer WebSphere eXtreme Scale WebSphere eXtreme Scale Client ou WebSphere eXtreme Scale for WebSphere Application Server pour l'offre de produit. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique.

Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses
Utilisez Installation Manager à l'aide d'un fichier de réponses pour installer les offres de produit WebSphere eXtreme Scale Client.

Rubrique parent : Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client

Java

Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

Utilisez Installation Manager à l'aide d'un fichier de réponses pour installer les offres de produit WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez installer les fichiers de produit nécessaires pour Installation Manager et avoir accès aux référentiels nécessaires. Pour plus d'informations, voir Installation d'IBM Installation Manager et des offres de produit WebSphere eXtreme Scale Client.

Pourquoi et quand exécuter cette tâche

A l'aide d'Installation Manager, vous pouvez utiliser des fichiers de réponses pour installer le produit de plusieurs manières. Vous pouvez enregistrer un fichier de réponses à l'aide de l'interface graphique.

Procédure

1. A partir d'une ligne de commande, accédez au sous-répertoire Eclipse du répertoire dans lequel vous avez installé Installation Manager.
2. Démarrez Installation Manager à partir de la ligne de commande en utilisant l'option d'enregistrement -record.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     IBMIM.exe -skipInstall "C:\temp\imRegistry"
       -record C:\temp\install_response_file.xml
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./IBMIM -skipInstall /var/temp/imRegistry
       -record /var/temp/install_response_file.xml
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./IBMIM -skipInstall racine_utilisateur/var/temp/imRegistry
       -record rép_base_utilisateur/var/temp/install_response_file.xml
     ```

   Conseil : Lorsque vous enregistrez un nouveau fichier de réponses, vous pouvez sélectionner le paramètre -skipInstall. Celui-ci vous offrira les avantages suivants :

   - Aucun fichier n'est installé, ce qui accélère l'enregistrement.
   - Si vous utilisez un emplacement de données temporaire avec le paramètre -skipInstall, Installation Manager y placera le registre d'installation lors de l'enregistrement. Lorsque vous redémarrerez Installation Manager sans le paramètre -skipInstall, vous pourrez utiliser le fichier de réponses pour l'installation au lieu du vrai registre d'installation.

     L'opération -skipInstall ne doit pas être utilisée sur l'emplacement de données de l'agent en cours utilisé par Installation Manager. Cette opération n'est pas prise en charge. Utilisez un nouvel emplacement inscriptible et utilisez à nouveau l'emplacement pour des sessions d'enregistrement ultérieures.

   Pour plus d'informations, voir le centre de documentation d'IBM Installation Manager Version 1.5.
3. Ajoutez les référentiels appropriés à vos préférences dans Installation Manager.
   a. Dans le menu du haut, cliquez sur **File > Preferences**.
   b. Sélectionnez **Repositories**
   c. Effectuez les actions suivantes pour chaque référentiel :
      i. Cliquez sur **Add Repository**.
      ii.
      iii. Entrez le chemin d'accès au fichier `repository.config` du référentiel Web éloigné ou du répertoire local dans lequel vous avez décompressé les fichiers du référentiel.

           Par exemple :

- Référentiels éloignés :

  ```
  https://downloads.mycorp.com:8080/WXS_85_repository
  ```

- Référentiels locaux :
  - Windows `C:\repositories\wxs85\local-repositories`
  - UNIX Linux `/var/repositories/wxs85/local-repositories`

iv. Cliquez sur **OK**.
v. Cliquez sur **Appliquer**.
vi. Cliquez sur **OK**.

d. Cliquez sur **Install**.

Remarque : Si vous êtes invité à vous authentifier, utilisez l'ID et le mot de passe IBM avec lesquels vous vous êtes enregistré sur le site Web du programme.

Installation Manager analyse ses référentiels définis à la recherche des packages disponibles.

4. Sélectionnez l'une des offres de produit suivantes ainsi que la version appropriée :
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

   Si le produit est déjà installé sur votre système, un message s'affichera pour vous en informer. Pour créer une autre installation du produit dans un autre emplacement, cliquez sur **Continuer**.

   Conseil : Si l'option **Search service repositories during installation and updates** est sélectionnée sur la page des préférences de référentiel dans Installation Manager et que vous êtes connecté à Internet, vous pouvez cliquer sur **Check for Other Versions and Extensions**. Ainsi, vous pouvez rechercher les mises à jour dans les référentiels de mise à jour par défaut pour les packages sélectionnés. Dans ce cas, il n'est pas nécessaire d'ajouter l'URL de référentiel de service spécifique à la page des préférences en matière de référentiel d'Installation Manager.

5. Sélectionnez les correctifs à installer.

   Les correctifs recommandés sont sélectionnés par défaut.

   Vous pouvez choisir de n'afficher que les correctifs recommandés et de masquer les autres.

6. Cliquez sur **Suivant**.
7. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
8. Indiquez le répertoire principal d'installation du produit.

   Le panneau affiche également le répertoire de ressources partagées et les informations d'espace disque.

   Remarque : La première fois que vous installez un package à l'aide d'Installation Manager, spécifiez le répertoire des ressources partagées. Le répertoire des ressources partagées est l'endroit où les artefacts d'installation se trouvent ; ils peuvent être utilisés par un plusieurs groupe de packages. Utilisez votre unité ayant la taille la plus importante pour cette installation. Vous ne pouvez pas changer l'emplacement du répertoire tant que vous n'avez pas désinstallé tous les packages.

   Restrictions :

   - Si vous supprimez l'emplacement cible par défaut et que vous ne renseignez pas une zone du répertoire d'installation, vous ne pouvez pas continuer.
   - N'utilisez pas de liens symboliques comme répertoire de destination.

     car ils ne sont pas pris en charge.
   - Windows La longueur maximale du chemin d'accès sur les systèmes d'exploitation Windows Server 2008, Windows Vista et Windows 7 est de 60 caractères.

9. Cliquez sur **Suivant**.
10. Sélectionnez les langues pour lesquelles un contenu traduit doit être installé.

    L'anglais est toujours sélectionné.

11. Cliquez sur **Suivant**.

12. Sélectionnez les fonctions que vous voulez installer.
    - Console

      Disponible pour toutes les offres de produit WebSphere eXtreme Scale. Vous pouvez choisir d'installer la console de surveillance. Avec la console Web, vous pouvez générer des graphiques des statistiques actuelles et historiques. Cette console fournit un certain nombre de graphiques pour des présentations générales et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.

    - Samples

      Disponible pour toutes les offres de produit WebSphere eXtreme Scale.

13. Cliquez sur **Suivant**.
14. Lisez le récapitulatif et cliquez sur **Install**.
    - Si l'installation aboutit, le programme affiche un message indiquant que l'installation a abouti.

      Remarque : Le programme peut également afficher d'importantes instructions post-installation.

    - Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
15. Cliquez sur **Finish**.
16. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Création d'un fichier de clés
Après avoir utilisé Installation Manager pour enregistrer un fichier de réponses afin d'installer des offres de produit WebSphere eXtreme Scale WebSphere eXtreme Scale Client, vous pouvez choisir de créer un fichier de clés. Si vous utilisez un référentiel distant nécessitant une authentification, vous pouvez créer un fichier clés pour l'installation.

Rubrique parent : Installation d'IBM Installation Manager à l'aide de fichiers de réponses

Tâches associées:
Désinstallation de WebSphere eXtreme Scale Client à l'aide de fichiers de réponses

Java

Création d'un fichier de clés

Après avoir utilisé Installation Manager pour enregistrer un fichier de réponses afin d'installer des offres de produit WebSphere eXtreme Scale WebSphere eXtreme Scale Client, vous pouvez choisir de créer un fichier de clés. Si vous utilisez un référentiel distant nécessitant une authentification, vous pouvez créer un fichier clés pour l'installation.

Avant de commencer

Vous devez enregistrer un fichier de réponses. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses.

Procédure

1. A partir d'une ligne de commande, accédez au sous-répertoire Eclipse du répertoire dans lequel vous avez installé Installation Manager.
2. Démarrez Installation Manager à partir de la ligne de commande en utilisant l'option d'enregistrement -record.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     IBMIM.exe -skipInstall "C:\temp\imRegistry"
       -keyring C:\IM\im.keyring
       -record C:\temp\keyring_response_file.xml
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./IBMIM -skipInstall /var/temp/imRegistry
       -keyring /var/IM/im.keyring
       -record /var/temp/keyring_response_file.xml
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./IBMIM -skipInstall racine_utilisateur/var/temp/imRegistry
       -keyring rép_base_utilisateur/var/IM/im.keyring
       -record racine_utilisateur/var/temp/keyring_response_file.xml
     ```

3. Lorsqu'une fenêtre s'affiche, vous demandant vos données d'identification pour le référentiel distant authentifié, entrez-les et **enregistrez**-les.
4. Cliquez sur **File > Exit** pour fermer Installation Manager.

   Pour plus d'informations, voir le centre de documentation d'IBM® Installation Manager Version 1.5.

Rubrique parent : Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Utilisez IBM® Installation Manager pour désinstaller des offres de produit WebSphere eXtreme Scale Client.

Désinstallation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique
Vous pouvez utiliser la console de l'assistant d'IBM Installation Manager pour désinstaller WebSphere eXtreme Scale Client.

Désinstallation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande
Vous pouvez désinstaller WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager à partir de la ligne de commande.

Désinstallation de WebSphere eXtreme Scale Client à l'aide de fichiers de réponses
Vous pouvez désinstaller WebSphere eXtreme Scale Client avec IBM Installation Manager à l'aide de fichiers de réponses.

Rubrique parent : Installation de WebSphere eXtreme Scale Client

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Vous pouvez utiliser la console de l'assistant d'IBM® Installation Manager pour désinstaller WebSphere eXtreme Scale Client.

Avant de commencer

Vous devez supprimer l'argument WebSphere eXtreme Scale de tous les profils WebSphere Application Server avant de désinstaller WebSphere eXtreme Scale. Vous ne pourrez pas exécuter l'annulation d'extension après avoir désinstallé WebSphere eXtreme Scale. Utilisez la commande manageprofiles pour réduire les profils existants dans un environnement WebSphere eXtreme Scale. .

Procédure

1. Désinstallez le produit.
    a. Arrêtez tous les processus en cours d'exécution dans votre environnement.
        - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
    b. Démarrez Installation Manager.
    c. Cliquez sur **Désinstaller**.
    d. Dans la fenêtre **Désinstaller des packages**, effectuez les actions suivantes.
        i. Sélectionnez l'une des offres de produit suivantes ainsi que la version appropriée :
            - WebSphere eXtreme Scale Client dans un environnement autonome
            - WebSphere eXtreme Scale Client for WebSphere Application Server Version 6
            - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
            - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
        ii. Cliquez sur **Next**.
    e. Si l'assistant de désinstallation affiche la liste des profils étendus, WebSphere Application Server, vous devez annuler l'extension de ces profils pour pouvoir poursuivre la désinstallation.
    f. Examinez les informations récapitulatives.
    g. Cliquez sur **Désinstaller**.
        - Si la désinstallation a réussi, le programme affichera un message pour confirmer le succès de l'opération.
        - Dans le cas contraire, cliquez sur **View log** pour corriger l'erreur.
    h. Cliquez sur **Terminer**.
    i. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.
2. Facultatif : Désinstallez IBM Installation Manager.

    Important : Avant de pouvoir désinstaller IBM Installation Manager, vous devez désinstaller tous les packages installés par Installation Manager.

    Pour plus d'informations sur cette procédure, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

Rubrique parent : Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Vous pouvez désinstaller WebSphere eXtreme Scale Client à l'aide d'IBM® Installation Manager à partir de la ligne de commande.

Avant de commencer

Vous devez supprimer l'argument WebSphere eXtreme Scale de tous les profils WebSphere Application Server avant de désinstaller WebSphere eXtreme Scale. Vous ne pourrez pas exécuter l'annulation d'extension après avoir désinstallé WebSphere eXtreme Scale. Utilisez la commande manageprofiles pour réduire les profils existants dans un environnement WebSphere eXtreme Scale.

Procédure

1. Ouvrez une session sur votre système.
2. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
3. Accédez au sous-répertoire `eclipse/tools` dans le répertoire où vous avez installé Installation Manager.
4. Utilisez la commande **`imcl`** pour désinstaller le produit.

   Windows

   ```
   imcl.exe uninstall com.ibm.websphere.v85,ID_fonction_facultative
     -installationDirectory répertoire_installation
   ```

   UNIX Linux

   ```
   ./imcl uninstall com.ibm.websphere.v85,ID_fonction_facultative
     -installationDirectory répertoire_installation
   ```

   Conseils :
   - *`ID_offre`* correspond à l'ID offre répertorié dans la rubrique ID d'offre pour les produits WebSphere eXtreme Scale Client.
   - Vous pouvez supprimer une liste de fonctions séparées par des virgules (ID fonction). Par exemple,

     ```
     imcl uninstall com.ibm.websphere.WXS.v85,xs.console.feature,xs.samples.feature
     ```

     - `client` indique la fonction client autonome
     - `server` indique la fonction serveur autonome
     - `console` indique la console de surveillance Web
     - `samples` indique les exemples
   - Si aucune liste de fonctions n'est spécifiée, la totalité du produit est désinstallée.

   Pour plus d'informations, accédez au d'centre de documentation d'IBM Installation Manager Version 1.5.
5. Si le processus de désinstallation affiche la liste des profils WebSphere Application Server étendus, vous devez annuler l'extension de ces profils pour pouvoir poursuivre la désinstallation.
6. Facultatif : Désinstallez IBM Installation Manager.

   Important : Avant de pouvoir désinstaller IBM Installation Manager, vous devez désinstaller tous les packages installés par Installation Manager.

   Pour plus d'informations sur l'utilisation du script de désinstallation pour exécuter cette procédure, consultez le centre de documentation d'IBM Installation Manager version 1.5.

Rubrique parent : Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande

Java

Désinstallation de WebSphere eXtreme Scale Client à l'aide de fichiers de réponses

Vous pouvez désinstaller WebSphere eXtreme Scale Client avec IBM® Installation Manager à l'aide de fichiers de réponses.

Avant de commencer

Vous devez supprimer l'argument WebSphere eXtreme Scale de tous les profils WebSphere Application Server avant de désinstaller WebSphere eXtreme Scale. Vous ne pourrez pas exécuter l'annulation d'extension après avoir désinstallé WebSphere eXtreme Scale. Utilisez la commande manageprofiles pour réduire les profils existants dans un environnement WebSphere eXtreme Scale.

Facultatif : Effectuez ou enregistrez l'installation d'Installation Manager et l'installation du produit dans un registre d'installation temporaire sur l'un de vos systèmes de manière à pouvoir enregistrer la désinstallation dans ce registre temporaire et non dans le registre standard ou est installé Installation Manager.

Pourquoi et quand exécuter cette tâche

Avec Installation Manager, vous pouvez utiliser des fichiers de réponses pour désinstaller le produit de plusieurs manières. Vous pouvez enregistrer un fichier de réponses à l'aide de l'interface graphique tel que décrit dans la procédure suivante, ou bien vous pouvez générer un nouveau fichier de réponses à la main ou en modifiant un exemple .

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
2. Facultatif : **Enregistrez un fichier de réponse pour désinstaller le produit :** Sur l'un de vos systèmes, effectuez les actions suivantes pour enregistrer un fichier de réponse qui vous permettra de désinstaller le module de fonctions:
   a. A partir d'une ligne de commande, passez au sous-répertoire eclipse du répertoire dans lequel vous avez installé Installation Manager.
   b. Démarrez Installation Manager à partir de la ligne de commande en utilisant l'option d'enregistrement -record.

      Par exemple :

      - Windows **Administrateur ou non administrateur :**

        ```
        IBMIM.exe -skipInstall "C:\temp\imRegistry"
          -record C:\temp\uninstall_response_file.xml
        ```

      - UNIX Linux **Administrateur :**

        ```
        ./IBMIM -skipInstall /var/temp/imRegistry
          -record /var/temp/uninstall_response_file.xml
        ```

      - UNIX Linux **Non administrateur :**

        ```
        ./IBMIM -skipInstall racine_utilisateur/var/temp/imRegistry
          -record racine_utilisateur/var/temp/uninstall_response_file.xml
        ```

Conseil : Si vous choisissez d'utiliser le paramètre -skipInstall avec un registre d'installation temporaire créé en suivant les instructions *Avant de commencer*, Installation Manager utilise le registre d'installation temporaire lors de l'enregistrement du fichier de réponses. Il est important de savoir que dès lors que le paramètre -skipInstall est spécifié, aucun package de fonction n'est installé ou désinstallé. Toutes les actions effectuées dans Installation Manager mettent tout simplement à jour les données d'installation stockées dans le registre temporaire indiqué. Une fois que le fichier de réponses est généré, il peut être utilisé pour désinstaller le produit : il supprime les fichiers du produit et met à jour le registre d'installation standard.

L'opération -skipInstall ne doit pas être utilisée sur l'emplacement de données de l'agent en cours utilisé par Installation Manager. Cette opération n'est pas prise en charge. Utilisez un nouvel emplacement inscriptible et utilisez à nouveau l'emplacement pour des sessions d'enregistrement ultérieures.

Pour plus d'informations, voir le centre de documentation d'IBM Installation Manager Version 1.5.

c. Cliquez sur **Désinstaller**.
d. Dans la fenêtre **Désinstaller des packages**, effectuez les actions suivantes.
    i. Sélectionnez l'une des offres de produit suivantes ainsi que la version appropriée :
        - WebSphere eXtreme Scale Client dans un environnement autonome
        - WebSphere eXtreme Scale Client for WebSphere Application Server Version 6
        - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
        - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
    ii. Cliquez sur **Next**.
    iii. Cliquez sur **Suivant**.
e. Examinez les informations récapitulatives.
f. Cliquez sur **Désinstaller**.
    - Si la désinstallation a réussi, le programme affichera un message pour confirmer le succès de l'opération.
    - Dans le cas contraire, cliquez sur **View log** pour corriger l'erreur.
g. Cliquez sur **Terminer**.
h. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

3. **Utilisez le fichier de réponses pour désinstaller le produit :** A partir d'une ligne de commande sur chacun des systèmes desquels vous souhaitez désinstaller le produit, accédez au sous-répertoire `eclipse/tools` du répertoire dans lequel vous avez installé Installation Manager et utilisez le fichier de réponses que vous avez créé pour procéder à la désinstallation du produit.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     imcl.exe
       input C:\temp\uninstall_response_file.xml
       -log C:\temp\uninstall_log.xml
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./imcl
       input /var/temp/uninstall_response_file.xml
       -log /var/temp/uninstall_log.xml
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./imcl
       input rép_base_utilisateur/var/temp/uninstall_response_file.xml
       -log racine_utilisateur/var/temp/uninstall_log.xml
     ```

   Pour plus d'informations, voir le centre de documentation d'IBM Installation Manager Version 1.5.

4. Facultatif : Répertoriez tous les packages installés pour vérifier la désinstallation.

   UNIX Linux

   ```
   ./imcl listInstalledPackages
   ```

   Windows

   ```
   imcl listInstalledPackages
   ```

5. Si le processus de désinstallation affiche la liste des profils WebSphere Application Server étendus, vous devez annuler l'extension de ces profils pour pouvoir poursuivre la désinstallation.
6. Facultatif : Désinstallez IBM Installation Manager.

   Important : Avant de pouvoir désinstaller IBM Installation Manager, vous devez désinstaller tous les packages installés par Installation Manager.

Pour plus d'informations sur l'utilisation du script de désinstallation pour exécuter cette procédure, accédez au centre de documentation d'IBM Installation Manager version 1.5.

Rubrique parent : Désinstallation de WebSphere eXtreme Scale Client à l'aide d'IBM Installation Manager

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

.NET

Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Vous pouvez installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution ou dans un environnement d'exécution et de production.

Pour générer et tester vos propres applications .NET, installez WebSphere eXtreme Scale Client for .NET dans votre environnement de développement. L'installation de l'environnement de développement inclut toujours l'installation de l'environnement d'exécution. Les assemblages d'exécution sont installés sur le disque et dans le cache GAC (Global Assembly Cache). L'installation de l'environnement de développement installe en outre du code exemple, l'intégration Visual Studio IntelliSense (pour les descriptions de classe et de méthode fly-over) et une documentation d'API. L'exemple de code source WebSphere eXtreme Scale Client for .NET et le projet Visual Studio sont installés dans le répertoire *net_client_home*\sample, et la documentation d'API est installée dans le répertoire *net_client_home*\doc.

.NET 2.5+ **Installation de WebSphere eXtreme Scale Client for .NET**
Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET si vous possédez des applications qui s'exécutent dans cette structure.

.NET 2.5+ **Installation de WebSphere eXtreme Scale Client for .NET en mode silencieux**
Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET en mode silencieux si vous ne souhaitez pas afficher le déroulement de l'installation ou devez installer le produit sur plusieurs machines. L'installation en mode silencieux signifie que vous devez enregistrer d'abord un fichier de réponses et les paramètres transmis à ce fichier.

.NET 2.5+ **Désinstallation de WebSphere eXtreme Scale Client for .NET**
Pour supprimer WebSphere eXtreme Scale Client for .NET dans votre environnement, vous pouvez le désinstaller depuis le panneau de configuration Windows ou enregistrer un fichier de réponses pour le désinstaller en mode silencieux. Il est préférable d'enregistrer un fichier de réponses lorsque vous avez plusieurs installations de WebSphere eXtreme Scale Client et que vous voulez supprimer rapidement ces installations.

.NET 2.5+ **Comment installer WebSphere eXtreme Scale Client for .NET sans utiliser le programme d'installation**
Si vous n'êtes pas autorisé à exécuter le fichier setup.exe, vous pouvez copier les fichiers d'une installation existante sur un autre système Windows.

Rubrique parent : Installation de WebSphere eXtreme Scale Client

.NET

Installation de WebSphere eXtreme Scale Client for .NET

Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET si vous possédez des applications qui s'exécutent dans cette structure.

Avant de commencer

- Installez WebSphere eXtreme Scale Client for .NET à partir du DVD. Le fichier `setup.exe` se trouve dans le répertoire racine, `/ClientForDotNet/setup.exe` (vous pouvez aussi le télécharger à partir du Site de support).
- Si vous comptez installer WebSphere eXtreme Scale Client for .NET dans un environnement de développement, vous devez utiliser un système Windows qui respecte la configuration matérielle et logicielle requise. Pour plus d'informations, voir Remarques relatives à Microsoft .NET.

Pourquoi et quand exécuter cette tâche

Vous pouvez installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution uniquement, ou dans un environnement d'exécution et dans un environnement de développement.

Procédure

1. Utilisez l'assistant pour installer le client à partir du DVD. Le fichier `setup.exe` se trouve dans le répertoire racine, `/ClientForDotNet/setup.exe`, ou vous pouvez le télécharger à partir du Site de support.
2. Exécutez le fichier `setup.exe`.
3. Suivez les invites de l'assistant et cliquez sur **Suivant** pour accéder à la page du type de configuration.
4. Si vous avez décidé d'installer WebSphere eXtreme Scale Client for .NET dans un environnement d'exécution, cliquez sur **Environnement d'exécution**, puis procédez comme suit :

   Cliquez sur **Installer** pour exécuter le programme d'installation, puis cliquez sur **Terminer**. Le répertoire d'installation par défaut est `C:\Program Files (x86)\IBM\WebSphere\eXtreme Scale .NET Client`
5. Si vous avez décidé d'installer WebSphere eXtreme Scale Client dans un environnement d'exécution et dans un environnement de développement, choisissez l'installation **Personnalisée** et procédez comme suit :
   a. Installez WebSphere eXtreme Scale Client dans le répertoire d'installation par défaut ou choisissez votre propre répertoire d'installation et cliquez sur **Suivant**.
   b. Par défaut, l'environnement d'exécution et l'environnement de développement sont sélectionnés. Vérifiez que vous disposez d'un espace disque suffisant pour installer les deux environnements. Cliquez sur **Suivant**.
   c. Choisissez l'emplacement des fichiers journaux et cliquez sur **Suivant**.
   d. Cliquez sur **Installer** pour exécuter le programme d'installation, puis cliquez sur **Terminer**.

Que faire ensuite

- Testez votre WebSphere eXtreme Scale Client for .NET en exécutant l'application d'initiation. Pour plus d'informations, voir Tutoriel : Démarrer avec des applications de grille de données simples.
- Configurez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de WebSphere eXtreme Scale Client for .NET.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
2.5+ Désinstallation de WebSphere eXtreme Scale Client for .NET
.NET 2.5+ Comment installer WebSphere eXtreme Scale Client for .NET sans utiliser le programme d'installation

.NET

Installation de WebSphere eXtreme Scale Client for .NET en mode silencieux

Vous pouvez installer WebSphere eXtreme Scale Client dans un environnement .NET en mode silencieux si vous ne souhaitez pas afficher le déroulement de l'installation ou devez installer le produit sur plusieurs machines. L'installation en mode silencieux signifie que vous devez enregistrer d'abord un fichier de réponses et les paramètres transmis à ce fichier.

Avant de commencer

- Procurez-vous WebSphere eXtreme Scale Client sur DVD. Le fichier `setup.exe` se trouve dans le répertoire racine, `/net_client_home/setup.exe`, ou vous pouvez le télécharger à partir du Site de support.
- Si vous envisagez d'installer WebSphere eXtreme Scale Client for .NET dans un environnement de développement, vous devez utiliser un système Windows qui respecte la configuration matérielle et logicielle décrite dans les remarques relatives à Microsoft .NET. Pour plus d'informations, voir Remarques relatives à Microsoft .NET.

Pourquoi et quand exécuter cette tâche

WebSphere eXtreme Scale Client for .NET peut être installé dans un environnement d'exécution ou dans un environnement d'exécution et de développement.

Procédure

1. Ouvrez une invite de commande et exécutez le script suivant : `setup.exe /r /f1"<Response_Files_Directory>\Setup.iss"` `<Response_Files_Directory>` est l'emplacement où vous voulez créer le fichier de réponses.
2. Suivez les invites de l'assistant et cliquez sur **Suivant** pour accéder à la page du type de configuration.
3. En fonction des options que vous choisissez, vous pouvez transmettre les valeurs suivantes pour créer le fichier de réponses `Setup.iss` :
   - Choisissez d'installer WebSphere eXtreme Scale Client dans un environnement d'exécution ou choisissez une installation personnalisée. Une personnalisation personnalisée permet d'installer le produit dans les deux environnements.
   - Si vous avez décidé d'installer WebSphere eXtreme Scale Client dans un environnement d'exécution, cliquez sur **Environnement d'exécution** et procédez comme suit :
     a. Cliquez sur **Désinstaller** et cliquez sur **Terminer**. L'installation par défaut est `C:\Program Files (x86)\IBM\WebSphere\eXtreme Scale .NET Client`
   - Si vous avez décidé d'installer WebSphere eXtreme Scale Client dans un environnement et un environnement d'exécution, choisissez l'installation **personnalisée** et procédez comme suit :
     a. Installez WebSphere eXtreme Scale Client dans le répertoire d'installation par défaut ou choisissez votre propre répertoire d'installation. Cliquez sur **Suivant**.
     b. Sélectionnez un environnement d'exécution et un environnement de développement. Si vous souhaitez effectuer l'installation dans les deux environnements, vérifiez que vous disposez d'un espace disque suffisant. Cliquez sur **Suivant**
     c. Choisissez l'emplacement des fichiers journaux et cliquez sur **Suivant**.
4. Cliquez sur **Désinstaller** et cliquez sur **Terminer**.
5. Ouvrez une invite de commande et exécutez le script suivant pour installer WebSphere eXtreme Scale Client en mode silencieux : `setup.exe /s /f1"<Response_Files_Directory>\Setup.iss"`, où `<Response_Files_Directory>` est l'emplacement où réside votre fichier de réponses.

Que faire ensuite

Vous pouvez mettre à jour ou modifier `SimpleClient` pour tester les API du client eXtreme Scale. Recherchez `SimpleClient` dans `<installation_directory>\sample\SimpleClient` dans le répertoire d'installation, puis chargez ce fichier dans Visual Studio afin d'afficher l'exemple d'application qui utilise des opérations simples de création, d'extraction, de mise à jour et de suppression. Utilisez `SimpleClient` comme guide pour accéder à la grille de données. Vous pouvez modifier cette application ou écrire une nouvelle application qui utilise le groupe d'API prises en charge du client eXtreme Scale for .NET. Pour plus d'informations, voir Tutoriel : Démarrer avec des applications de grille de données simples.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
2.5+ Désinstallation de WebSphere eXtreme Scale Client for .NET

.NET

Désinstallation de WebSphere eXtreme Scale Client for .NET

Pour supprimer WebSphere eXtreme Scale Client for .NET dans votre environnement, vous pouvez le désinstaller depuis le panneau de configuration Windows ou enregistrer un fichier de réponses pour le désinstaller en mode silencieux. Il est préférable d'enregistrer un fichier de réponses lorsque vous avez plusieurs installations de WebSphere eXtreme Scale Client et que vous voulez supprimer rapidement ces installations.

Avant de commencer

Si vous souhaitez désinstaller le produit dans un environnement de développement, veillez à arrêter Visual Studio.

Avertissement : Le programme de désinstallation supprime tous les fichiers binaires et la maintenance, telle que les groupes de correctifs et les correctifs temporaires, en même temps.

Procédure

1. Arrêtez les processus .NET eXtreme Scale.
2. Vous pouvez désinstaller WebSphere eXtreme Scale Client for .NET au moyen de l'une des méthodes suivantes :
   - Effectuez la désinstallation à partir du panneau de configuration Windows, cliquez sur Ajouter ou supprimer des programmes, puis sélectionnez IBM WebSphere eXtreme Scale Client for .NET.
   - Si vous voulez enregistrer un fichier de réponses, ouvrez une invite de commande et exécutez le script suivant :

     ```
     setup.exe /uninst /r /f1"<Response_Files_Directory>\Setup.iss"
     ```

     a. L'assistant de désinstallation s'ouvre et une fenêtre de confirmation s'affiche pour vérifier que vous voulez supprimer WebSphere eXtreme Scale Client for .NET et toutes ses fonctions. Cliquez sur **OK**.
     b. A la fin de la désinstallation, cliquez sur **Terminer**.
3. Facultatif : si vous voulez utiliser votre fichier de réponses pour désinstaller une installation existante de WebSphere eXtreme Scale Client, exécutez votre fichier de réponses comme suit :

   a. Ouvrez une invite de commande et exécutez le script suivant pour désinstaller WebSphere eXtreme Scale Client for .NET en mode silencieux :

   ```
   setup.exe /uninst /s /f1"<Response_Files_Directory>\Setup.iss"
   ```

Que faire ensuite

Vérifiez Windows Explorer pour vous assurer que tous les dossiers ont été supprimés du répertoire d'installation. Vous devez également vérifier le panneau de configuration Windows pour vous assurer que le produit n'est plus répertorié. Le programme d'installation ne supprime pas les dossiers comportant des fichiers qui ont été générées après l'installation, par exemple les fichiers journaux, les fichiers de configuration personnalisés et les artefacts créés par la génération de l'exemple `SimpleClient`.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET 2.5+ Installation de WebSphere eXtreme Scale Client for .NET
.NET 2.5+ Installation de WebSphere eXtreme Scale Client for .NET en mode silencieux

.NET

Comment installer WebSphere eXtreme Scale Client for .NET sans utiliser le programme d'installation

2.5+ Si vous n'êtes pas autorisé à exécuter le fichier `setup.exe`, vous pouvez copier les fichiers d'une installation existante sur un autre système Windows.

Avant de commencer

Vous devez disposer d'une installation opérationnelle de WebSphere eXtreme Scale Client for .NET à partir de laquelle vous pourrez copier les fichiers. Pour plus d'informations, voir Présentation de l'installation de WebSphere eXtreme Scale Client for .NET.

Pourquoi et quand exécuter cette tâche

Si vous n'êtes pas autorisé à exécuter le fichier `setup.exe` sur votre serveur client, vous pouvez copier les fichiers d'une installation existante de WebSphere eXtreme Scale Client for .NET dans le répertoire d'exécution de votre application .NET.

Avertissement : Les fonctionnalités de migration et de mise à niveau automatiques ne sont disponibles que dans le programme d'installation de WebSphere eXtreme Scale Client for .NET. Si vous utilisez la procédure ci-dessous pour installer le produit, vous devrez donc effectuer manuellement les migrations et les mises à niveau.

Procédure

1. Windows A partir de votre installation de WebSphere eXtreme Scale Client for .NET, copiez les fichiers ci-dessous sur le système Windows cible. Le répertoire dans lequel vous effectuez la copie sur le système Windows (répertoire cible) doit être le répertoire d'exécution du processus qui exécute l'application .NET pour votre grille de données. Ce processus doit disposer d'un accès en lecture et en écriture au répertoire cible. Les fichiers journaux sont créés dans un dossier `logs` dans ce répertoire.
   - *net_client_home*`\bin\IBM.WebSphere.Caching.dll`
   - *net_client_home*`\bin\IBM.WebSphere.Caching.CredentialGenerator.dll` (ce fichier ne doit être copié que si l'application WebSphere eXtreme Scale Client for .NET est configurée pour utiliser le générateur de données d'identification fourni)
   - *net_client_home*`\config\Client.Net.properties`
   - *net_client_home*`\config\Client.Net.Log.config`
2. Facultatif : Installez manuellement l'assemblage `IBM.WebSphere.Caching.dll` dans le cache GAC (Global Assembly Cache). La procédure à suivre dépend de votre environnement Windows.

Que faire ensuite

Configurez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de WebSphere eXtreme Scale Client for .NET.

Rubrique parent : .NET 2.5+ Présentation de l'installation de WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET 2.5+ Installation de WebSphere eXtreme Scale Client for .NET

Java

Installation de profil Liberty

Vous installez l'environnement de service d'applications profil Liberty en utilisant le gestionnaire d'installation ou en exécutant un fichier d'archive Java™ (fichier JAR).

Pourquoi et quand exécuter cette tâche

Pour installer le profil Liberty, vous pouvez installer le WebSphere eXtreme Scale Client pour WebSphere Application Server version 8.5 ou ultérieur, ou exécuter un fichier JAR fourni..

Sur les systèmes z/OS, le profil Liberty fournit un environnement d'exécution. Vous pouvez utiliser cet environnement en mode natif à l'aide de la console MVS. Pour le développement d'applications, vous pouvez envisager d'utiliser les outils basés sur la technologie Eclipse sur un système réparti distinct, un système d'exploitation Mac OS, ou dans un shell Linux sur un système z/OS.

Procédure

- Installez le profil Liberty de WebSphere Application Server de l'une des manières suivantes :
  - Dans WebSphere DataPower XC10 Appliance, vous devez installer le WebSphere eXtreme Scale Client. Pour pouvoir exécuter le profil Liberty dans cette topologie, vous devez Installer l'environnement de service d'applications profil Liberty en exécutant un fichier d'archive Java (fichier JAR).
- Installez WebSphere eXtreme Scale Client version 8.5 ou ultérieur. Lorsque vous installez le produit de cette manière, le profil Liberty est automatiquement installé.
- Faites migrer votre environnement de profil Liberty. Lorsque vous migrez d'une édition majeure du profil Liberty vers une édition majeure ultérieure, vous devez modifier les numéros de version de fonctionnalité dans le fichier `server.xml` de votre profil Liberty.

  Par exemple, dans la version initiale du profil Liberty prise en charge par le produit, les numéros de version de fonctionnalité étaient au niveau `1.0`. Dans WebSphere eXtreme Scale Client version 8.6 et ultérieur, ces numéros sont au niveau `1.1`.

  Remarque : Lorsque vous effectuez une mise à niveau vers WebSphere Application Server version 8.5.5, le profil Liberty est supprimé. WebSphere eXtreme Scale prend en charge la suppression de cette fonctionnalité. Cependant, si vous revenez ensuite à WebSphere Application Server version 8.5, le profil Liberty est de nouveau ajouté, ce qui crée des problèmes dans WebSphere eXtreme Scale.

Profil Liberty et mise en cache des données
Vous pouvez utiliser des produits de mise en cache des données avec le profil Liberty de WebSphere Application Server afin de développer des sessions HTTP ainsi que des connexions client-serveur utilisant la passerelle REST, et de gérer d'autres scénarios d'intégration de cache.

Installation de l'environnement de service d'applications profil Liberty en exécutant un fichier JAR
En exécutant le fichier d'archive Java (JAR) qui contient l'image de distribution, vous installez l'environnement de service d'applications et vous êtes prêt à créer un serveur Liberty.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Tâches associées:
Installation de WebSphere eXtreme Scale Client à l'aide de l'interface graphique
Installation de WebSphere eXtreme Scale Client à l'aide de la ligne de commande
Installation de WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses

Profil Liberty et mise en cache des données

Java Vous pouvez utiliser des produits de mise en cache des données avec le profil Liberty de WebSphere Application Server afin de développer des sessions HTTP ainsi que des connexions client-serveur utilisant la passerelle REST, et de gérer d'autres scénarios d'intégration de cache.

Dans WebSphere DataPower XC10 Appliance, vous pouvez utiliser le profil Liberty pour établir une connexion à la grille de données du dispositif. Par exemple, lorsque vous installez WebSphere eXtreme Scale Client avec le profil Liberty, vous avez accès à des fonctions qui vous permettent de gérer les applications de session HTTP, les applications client Java et les applications client REST installées dans le profil Liberty.

Les fonctions ci-dessous contiennent des informations concernant les principales fonctions disponibles. L'inclusion d'une fonction dans la configuration peut entraîner le chargement automatique d'une ou plusieurs fonctions. Chaque fonction comporte une brève description et un exemple de la façon dont cette fonction est déclarée.

Fonction client

La fonction client contient la plus grande partie du modèle de programmation pour eXtreme Scale. Ajoutez la fonction client lorsqu'une application qui s'exécute dans le profil Liberty utilisera des API eXtreme Scale.

Fichier *racine_installation_wlp*/usr/server/*racine_installation_wlp*/server.xml

```
<server description="WebSphere eXtreme Scale Client">

  <featureManager>
    <feature>eXtremeScale.client-1.1</feature>
  </featureManager>

 </server>
```

Fonction Web

La fonction Web est obsolète. Utilisez la fonction webApp pour répliquer les données de session HTTP pour la tolérance aux pannes.

La fonction Web permet d'étendre le profil Liberty à une application Web. Ajoutez la fonction Web lorsque vous souhaitez répliquer des données de session HTTP dans le cadre de la tolérance aux pannes.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

  <featureManager>
    <feature>eXtremeScale.web-1.1</feature>
  </featureManager>

  <xsWebAppV85/>
</server>
```

Fonction webApp

La fonction webApp permet d'étendre l'application Web du profil Liberty. Ajoutez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.webApp-1.1</feature>
</featureManager>

<xsWebApp/>
</server>
```

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">
```

```
<featureManager>
<feature>eXtremeScale.webGrid-1.1</feature>
</featureManager>

<xsWebGrid/>
</server>
```

Fonction REST

Utilisez la passerelle REST (Representational State Transfer) pour accéder aux grilles de données simples hébergées par un collectif dans le profil profil Liberty.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.rest-1.1</feature>
</featureManager>

<xsRest/>
</server>
```

Fonction de cache dynamique

Un serveur profil Liberty peut héberger une grille de données qui met en mémoire cache les données des applications pour lesquelles la mémoire cache dynamique est activée.

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

  <featureManager>
    <feature>eXtremeScale.dynacacheGrid-1.1</feature>
  </featureManager>

  <xsDynacacheGrid/>
</server>
```

2.5+

Fonction de cache dynamique

Le serveur de profil Liberty peut héberger WebSphere eXtreme Scale, que vous pouvez configurer comme fournisseur de cache dynamique à l'aide de cette fonction.

Voyez l'exemple suivant pour savoir comment utiliser le fournisseur de cache par défaut dans le profil Liberty:

Fichier *racine_installation_wlp*/usr/server/*nom_serveur*/server.xml

```
<server description="WebSphere eXtreme Scale enabled Web Server">

<featureManager>
<feature>eXtremeScale.dynacacheapp-1.1</feature>
</featureManager>

<xsClientDomain default="production">
<endpointConfig> production;localhost:2809 </endpointConfig>
</xsClientDomain>

<distributedMap id="baseCache" libraryRef="idgenerator" cacheProviderName="WebSphere
eXtreme Scale">
<xsDynacacheApp remoteDomain="production" />
</distributedMap>

<distributedMap id="cache01" jndiName="cache01" memorySizeInEntries="2000"
createCacheAtServerStartup="true" cacheProviderName="WebSphere eXtreme Scale">
<xsDynacacheApp remoteDomain="production" />
```

```
</distributedMap>
</server>
```

Rubrique parent : Installation de profil Liberty

Java

Installation de l'environnement de service d'applications profil Liberty en exécutant un fichier JAR

En exécutant le fichier d'archive Java (JAR) qui contient l'image de distribution, vous installez l'environnement de service d'applications et vous êtes prêt à créer un serveur Liberty.

Pourquoi et quand exécuter cette tâche

Vous pouvez installer l'environnement de service d'applications profil Liberty en exécutant un fichier JAR, comme indiqué dans cette rubrique ou en utilisant le gestionnaire d'installation.

Lorsque vous exécutez le fichier JAR pour WebSphere Application Server pour installer profil Liberty, vous devez d'abord extraire le fichier JAR. Ensuite, extrayez le fichier JAR profil Liberty pour WebSphere eXtreme Scale. Si vous utilisez IBM Installation Manager pour installer WebSphere Application Server version 8.5 et obtenir le profil Liberty, vous devez utiliser le gestionnaire d'installation pour installer également WebSphere eXtreme Scale.

Cette tâche prend en charge les éditions suivantes :

- WebSphere Application Server Liberty Core
- WebSphere Application Server, édifions Base et Developer
- WebSphere Application Server, Network Deployment
- WebSphere Application Server for z/OS

Pour les informations de téléchargement des environnements de gestion d'applications et de mise en cache des données profil Liberty, voir la page des téléchargements de la communauté WASdev.

Procédure

1. Extrayez l'image de distribution du profil Liberty WebSphere Application Server vers le répertoire de votre choix.

   Cette image est fournie sous la forme d'un fichier JAR ; par exemple, `wlp-`*`édition`*`-8.6.0.0.jar`. Extrayez ce fichier JAR de l'une des manières suivantes :
   - Pour extraire l'image de distribution en utilisant l'assistant, exécutez `java -jar wlp-`*`édition`*`-8.6.0.0.jar`.
   - Pour extraire l'image de distribution en acceptant les conditions de la licence et les conditions d'installation silencieuse, exécutez `java -jar wlp-édition-8.6.0.0.jar -acceptLicense`.
   - Pour afficher toutes les options disponibles, exécutez `java -jar wlp-édition-8.6.0.0.jar -help`.

   Tous les fichiers de serveur d'applications sont stockés dans le répertoire `wlp`.
2. Facultatif : Définissez la variable `JAVA_HOME` de votre environnement.

   Le profil Liberty nécessite un JRE pour son exécution. Il ne partage pas le JDK ou le JRE que le profil complet WebSphere Application Server utilise. Vous pouvez définir l'emplacement du JDK ou du JRE en utilisant la propriété **JAVA_HOME** dans le fichier `server.env`, comme indiqué dans Personnalisation de l'environnement du profil Liberty. Sur les systèmes Linux et UNIX, vous pouvez définir la variable **JAVA_HOME** dans le fichier `.bashrc` de l'utilisateur ou ajouter le chemin du JDK ou du JRE à la variable d'environnement **PATH**. Sur un système Windows, vous pouvez aussi définir **JAVA_HOME** en tant que variable d'environnement du système ou ajouter le chemin du JDK ou du JRE à la variable **PATH** du système.

   Windows Sur un système Windows, par exemple, utilisez les commandes suivantes pour faire pointer la propriété **JAVA_HOME** sur l'emplacement où votre JDK est installé et pour ajouter le répertoire `/bin` de votre installation Java™ à la variable PATH :

   ```
   set JAVA_HOME=C:\Progra~1\Java\JDK16
   set PATH=%JAVA_HOME%\bin;%PATH%
   ```

   Remarques :
   - L'environnement d'exécution profil Liberty recherche la commande **java** dans l'ordre suivant : propriété **JAVA_HOME**, propriété **JRE_HOME** et propriété système **PATH**.
   - Pour plus d'informations sur les environnements pris en charge Java et savoir où vous les procurer, voir Niveaux Java minimaux pris en charge dans profil Liberty : Restrictions connues de l'environnement d'exécution.
3. Extrayez l'image de distribution WebSphere eXtreme Scale vers le répertoire dans lequel vous avez extrait le fichier `wlp-`*`édition`*`-8.6.0.0.jar`.

Cette image se trouve dans le fichier JAR `wxs-wlp_8.6.0.0.jar` en version 8.6 et `wxs-wlp_8.6.0.2.jar` en version 8.6 groupe de correctifs 2. Pour extraire l'image de distribution, exécutez le fichier JAR ; par exemple, exécutez la commande suivante, en fonction de votre version de eXtreme Scale:

```
java -jar wxs-wlp_8.6.0.0.jar -acceptLicense
```

```
java -jar wxs-wlp_8.6.0.2.jar -acceptLicense
```

Résultats

Si vous extrayez le fichier le fichier `wxs-wlp_8.6.0.0.jar` pour la version 8.6 et le fichier `wxs-wlp_8.6.0.2.jar` pour la version 8.6 groupe de correctifs 2, eXtreme Scale est installé par dessus le profil Liberty WebSphere Application Server lorsque vous extrayez les deux fichiers JAR vers le même répertoire.

Rubrique parent : Installation de profil Liberty

Identification et résolution des incidents liés à l'installation du produit

IBM® Installation Manager est un programme d'installation commun à de nombreux logiciels IBM qui vous permet d'installer cette version de WebSphere eXtreme Scale.

Résultats

Remarques sur la journalisation et le traçage :

- Pour consulter aisément les journaux, ouvrez le gestionnaire d'installation et cliquez sur **File > View Log**. Pour ouvrir un fichier journal individuel, il vous suffira alors de le sélectionner dans le tableau et de cliquer sur l'icône **Open log file**.
- Les journaux sont situés dans le répertoire `logs`, à l'emplacement des données applicatives d'Installation Manager. Exemple :
  - Windows **Installation par un administrateur :**

    ```
    C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
    ```

  - Windows **Installation par un non administrateur :**

    ```
    C:\Documents and Settings\nom_utilisateur\Application Data\IBM\Installation Manager
    ```

  - UNIX Linux **Installation par un administrateur :**

    ```
    /var/IBM/InstallationManager
    ```

  - UNIX Linux **Installation par un non administrateur :**

    ```
    rép_utilisateur/var/ibm/InstallationManager
    ```

- Les principaux fichiers journaux sont horodatés et conservés dans le répertoire `logs` au format XML. Ils peuvent donc être consultés à l'aide d'un navigateur Web standard.
- Le fichier `log.properties` du répertoire `logs` indique le niveau de journalisation ou de traçage appliqué par Installation Manager. Pour activer la fonction de traçage des plug-ins de WebSphere eXtreme Scale, par exemple, créez un fichier `log.properties` contenant les informations suivantes :

  ```
  com.ibm.ws=DEBUG
  com.ibm.cic.agent.core.Engine=DEBUG
  global=DEBUG
  ```

  Redémarrez Installation Manager au besoin. Installation Manager fournira ainsi les traces des plug-ins de WebSphere eXtreme Scale.

Remarques sur la résolution des problèmes :

- UNIX Linux Par défaut, certains systèmes HP-UX sont configurés pour ne pas utiliser de système DNS pour résoudre les noms d'hôte. Il est par conséquent possible qu'Installation Manager ne puisse pas se connecter à un référentiel externe.

  Vous pouvez vérifier la connexion au référentiel à l'aide de l'utilitaire Ping mais nslookup ne renverra aucun résultat.

  Demandez à votre administrateur système de configurer votre machine afin qu'elle utilise un système DNS ou utilisez l'adresse IP du référentiel.
- Dans certains cas, il peut être nécessaire d'ignorer les mécanismes de vérification existants dans Installation Manager.
  - Sur certains systèmes de fichiers réseau, l'espace disque peut ne pas être signalé correctement et il peut être nécessaire d'ignorer la vérification d'espace disque et de poursuivre l'installation.

    Pour désactiver la vérification d'espace disque, spécifiez la propriété système suivante dans le fichier `config.ini` du répertoire *`racine_install_IM`*`/eclipse/configuration` et redémarrez Installation Manager :

```
cic.override.disk.space=tailleunité
```

où *taille* est un chiffre entier et *unité* est laissé vide pour octets, a la valeur k pour kilo, m pour mégaoctets ou g pour gigaoctets. Par exemple :

```
cic.override.disk.space=120 (120 octets)
cic.override.disk.space=130k (130 kilooctets)
cic.override.disk.space=140m (140 mégaoctets)
cic.override.disk.space=150g (150 gigaoctets)
cic.override.disk.space=true
```

Installation Manager indique que la taille de l'espace disque est Long.MAX_VALUE. Au lieu d'afficher une grande quantité d'espace disque disponible, N/A s'affiche.

- Pour ignorer la vérification des prérequis pour le système d'exploitation, ajoutez `disableOSPrereqChecking=true` au fichier `config.ini` dans *racine_install_IM*`/eclipse/configuration` et redémarrez Installation Manager.

  Si vous devez utiliser l'une de ces méthodes, contactez le support IBM pour obtenir de l'aide et développer une solution n'impliquant pas d'ignorer les mécanismes de vérification d'Installation Manager.

- Pour plus d'informations sur l'utilisation d'Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

  Pour en savoir plus sur la dernière version d'Installation Manager, lisez les notes sur l'édition. Pour accéder aux notes sur l'édition, procédez comme suit :

  - Windows Cliquez sur **Démarrer > Programmes > IBM Installation Manager > Release Notes**.
  - UNIX Linux Accédez au sous-répertoire de la documentation dans le répertoire dans lequel Installation Manager est installé et ouvrez le fichier `readme.html`.

- Si une erreur fatale se produit lors de l'installation du produit, effectuez les étapes suivantes :
  - Faites une copie de sauvegarde de votre répertoire d'installation de produit actuel au cas où le service de support IBM aurait besoin de l'examiner ultérieurement.
  - Utilisez Installation Manager pour désinstaller tout ce que vous avez installé dans l'emplacement d'installation du produit (groupe de packages). Vous risquez de rencontrer des erreurs, mais vous pouvez les ignorer en toute sécurité.
  - Supprimez tout ce qui reste dans le répertoire d'installation du produit.
  - Utilisez Installation Manager pour réinstaller le produit dans le même emplacement ou dans un nouvel emplacement.

Remarque sur les informations de version et d'historique : Les commandes **`versionInfo`** et **`historyInfo`** retournent les informations de version et d'historique basées sur toutes les activités d'installation, de désinstallation, de mise à jour et de rétrogradation effectuées sur le système.

Rubrique parent : Installation de WebSphere DataPower XC10 Appliance

Mise à jour de WebSphere DataPower XC10 Appliance

Pour appliquer un pack de maintenance à votre environnement WebSphere DataPower XC10 Appliance, appliquez d'abord les éventuelles mises à jour du microprogramme. Vous pouvez ensuite appliquer des mises à jour à vos installations WebSphere eXtreme Scale Client.

1. Mise à jour du microprogramme
   Vous pouvez mettre à niveau le logiciel de votre IBM® WebSphere DataPower XC10 Appliance à l'aide des mises à jour du microprogramme. Téléchargez les nouvelles versions du microprogramme à partir du site Web Fix Central, puis mettez à jour le logiciel du dispositif.

2. Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server
   Lors de la migration de WebSphere Application Server vers une nouvelle version, vous pouvez également faire migrer la configuration de WebSphere eXtreme Scale Client vers la nouvelle installation WebSphere Application Server.

3. Installation des groupes de correctifs à l'aide d'IBM Installation Manager
   Vous pouvez utiliser IBM Installation Manager pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale Client. Les groupes de correctifs peuvent être installés à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

4. Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager
   Vous pouvez utiliser IBM Installation Manager pour rétrograder les offres de produit WebSphere eXtreme Scale Client vers une version précédente. Vous pouvez désinstaller les groupes de correctifs à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

5. 2.5+ Mise à niveau de WebSphere eXtreme Scale Client for .NET
   Pour mettre à niveau une installation existante de WebSphere eXtreme Scale Client for .NET, vous pouvez utiliser le programme d'installation. Ce dernier détecte l'installation existante et remplace les fichiers appropriés.

Mise à jour du microprogramme

Vous pouvez mettre à niveau le logiciel de votre IBM® WebSphere DataPower XC10 Appliance à l'aide des mises à jour du microprogramme. Téléchargez les nouvelles versions du microprogramme à partir du site Web Fix Central, puis mettez à jour le logiciel du dispositif.

Avant de commencer

Vous devez disposer des droits d'accès d'administrateur du dispositif.

Attendez que toutes les tâches actives soient terminées avant de lancer le processus de mise à jour du microprogramme. En effet, ce processus met fin à tous les travaux en cours d'exécution, ce qui pourrait entraîner alors une incohérence des données. Tenez-en compte avant de lancer le processus de mise à jour.

Important : Si vous installez le microprogramme pour la première fois sur un nouveau dispositif, vous devez exécuter la commande **`clear-all`** sur le dispositif une fois la mise à jour du microprogramme terminée. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Pourquoi et quand exécuter cette tâche

Vous pouvez mettre à jour le microprogramme du dispositif en téléchargeant une nouvelle mise à jour du microprogramme à partir du site IBM et en mettant à jour le dispositif avec le nouveau microprogramme. Vous pouvez exécuter la mise à jour du microprogramme dans l'interface utilisateur ou sur la ligne de commande.

ATTENTION :

Lors d'une mise à niveau vers la version 2.5 à partir de la version 2.0 ou d'une version antérieure, vous ne pouvez pas exécuter de mise à niveau de microprogramme sur une collectivité traitant une charge de travail. En effet, lors de la mise à niveau d'une collectivité, toutes les données chargées dans les grilles de données sont perdues. Quand vous exécutez des mises à niveau de microprogramme sur les dispositifs d'une collectivité, le dispositif mis à niveau ne redémarre pas en intégralité tant que tous les dispositifs de la collectivité ne sont pas mis à niveau. Les dispositifs restants qui ne sont pas encore mis à niveau prennent en charge les demandes. Vous devez terminer la mise à niveau du microprogramme d'un dispositif avant de démarrer le processus sur un autre dispositif de la collectivité. N'apportez pas de modifications de configuration tant que tous les dispositifs ne sont pas au même niveau de microprogramme.

Procédure

- **Téléchargez la mise à niveau du microprogramme :**

  Accédez au site IBM Fix Central et téléchargez une mise à jour du microprogramme dans votre système de fichiers local. Dans la page Fix Central, sélectionnez **WebSphere** comme **Groupe de produits** et **WebSphere DataPower XC10 Appliance** dans la liste des produits. Téléchargez la mise à jour de microprogramme appropriée pour votre type de matériel de dispositif :
  - Les mises à jour du microprogramme pour le type de dispositif 7199-92X correspondent à un fichier unique portant l'extension `.scrypt3`.

  Ces fichiers `scrypt` sont assortis d'une signature pour garantir l'intégrité de la mise à jour exécutée. Si vous utilisez l'interface utilisateur pour exécuter la mise à niveau du microprogramme, sauvegardez ce fichier sur l'ordinateur que vous utilisez pour accéder à l'interface utilisateur. Si vous utilisez l'interface de ligne de commande pour exécuter la mise à niveau du microprogramme, sauvegardez ce fichier sur un serveur auquel vous pouvez accéder à partir du dispositif.
- **Pour exécuter la mise à jour du microprogramme dans l'interface utilisateur :**
  1. Connectez-vous à l'interface utilisateur.
  2. Accédez au panneau Paramètres.
     - Dans l'interface utilisateur de WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres**.
     - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.
  3. Développez l'entrée **Microprogramme**. Le niveau du microprogramme installé sur le dispositif s'affiche.
  4. Installez une nouvelle mise à jour du microprogramme.
     a. Cliquez sur **Parcourir...** pour pouvoir sélectionner le fichier de mise à jour du microprogramme.
     b. Sélectionnez le fichier de mise à jour du microprogramme et cliquez sur **OK**.
     c. Cliquez sur **Mettre à niveau**. La durée du téléchargement de la mise à jour du

microprogramme dépend du débit de votre connexion. Un message s'affiche à l'issue du téléchargement pour vous aviser du démarrage de la mise à jour du microprogramme. Au début de cette mise à jour, le dispositif est redémarré sans que l'écran n'indique de progression ou n'affiche de modifications dans l'interface utilisateur. Les modifications ne sont pas affichées car votre session a pris fin au redémarrage du dispositif et l'interface utilisateur est indisponible au cours du processus de mise à niveau.

Important : Le dispositif redémarre plusieurs fois pendant le processus de mise à jour du microprogramme. N'interrompez pas le processus ou ne redémarrez pas manuellement le dispositif pendant la mise à jour du microprogramme.

Pour vérifier que la mise à jour du microprogramme s'est achevée, vous devez vous connecter à nouveau lorsque cette opération vous paraît terminée. La mise à jour du microprogramme dure en moyenne 10 à 15 minutes, mais elle peut prendre plus longtemps. Si vous souhaitez surveiller la progression de la mise à jour du microprogramme, utilisez le panneau d'état de démarrage de l'interface utilisateur du dispositif ou la commande **`start-progress`** de l'interface de ligne de commande. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.

- **Pour exécuter la mise à jour du microprogramme dans l'interface de ligne de commande :**

  1. Connectez-vous au dispositif à l'aide de la ligne de commande ou de la console série. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.
  2. Placez le fichier `.scrypt` sur le dispositif. A partir de l'interface de ligne de commande, exécutez la commande suivante :

     ```
     file get <url_fichier_scrypt> <fichier_microprogramme>
     ```

     Où *`url_fichier_scrypt`* représente au serveur local sur lequel vous avez sauvegardé le fichier `.scrypt` à partir de Fix Central, et *`fichier_microprogramme`* au nom du fichier `.scrypt` à utiliser sur le dispositif.
  3. Installez la mise à jour du microprogramme. Exécutez la commande suivante :

     ```
     firmware upgrade <fichier_microprogramme>
     ```

     Important : Pendant le processus de mise à niveau du microprogramme, le dispositif redémarre plusieurs fois. N'interrompez pas le processus ou ne redémarrez pas manuellement le dispositif pendant la mise à jour du microprogramme.
  4. Surveillez la progression du démarrage du dispositif.

     2.5+ Utilisez l'une des options suivantes :

     - 2.5+ Utilisez l'interface utilisateur du dispositif. Dans la barre d'adresse du navigateur Web, entrez l'URL et le port suivants : `https://<nom-hôte_dispositif>:9443/`. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.
     - Exécutez la commande **`start-progress`**. Lorsque cette commande renvoie `STARTED`, cela signifie que le dispositif est prêt à l'emploi.

Résultats

Le microprogramme du dispositif a été mis à jour. Pour vérifier si la mise à niveau du microprogramme a réussi, affichez le niveau actuel du microprogramme dans la section Microprogramme du panneau **Dispositif** > **Paramètres**. Ce panneau affiche également le type de modèle et le numéro de série du dispositif. Répétez ces étapes pour les autres dispositifs de la collectivité.

Que faire ensuite

Si vous installez le microprogramme pour la première fois sur un nouveau dispositif, vous devez exécuter la commande **`clear-all`** sur le dispositif une fois la mise à jour du microprogramme terminée. Pour plus d'informations, voir Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique suivante : Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server

Tâches associées:
Démarrage rapide : Installation du matériel du dispositif
Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance

Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server

Lors de la migration de WebSphere Application Server vers une nouvelle version, vous pouvez également faire migrer la configuration de WebSphere eXtreme Scale Client vers la nouvelle installation WebSphere Application Server.

Avant de commencer

- On part du principe que WebSphere eXtreme Scale Version 7 et WebSphere eXtreme Scale Version 8 sont installés sur le même serveur.
- Faites migrer WebSphere Application Server Version 7 vers WebSphere Application Server Version 8. Pour plus d'informations, voir Migration des configurations de produit.
- Installez WebSphere eXtreme Scale Client version 8 dans votre installation WebSphere Application Server version 8. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server. Tous les scripts de migration de WebSphere eXtreme Scale Client doivent être exécutés à partir de WebSphere eXtreme Scale Client version 8.5 ou ultérieure. Par exemple, pour la migration de la version 7.x vers la version 8, exécutez les scripts de migration à partir du répertoire `<racine_install_WXS_v8>/bin`.

Pourquoi et quand exécuter cette tâche

Lorsque vous installez une nouvelle version de WebSphere Application Server dotée de l'intégration WebSphere eXtreme Scale, mettez d'abord à niveau WebSphere Application Server à l'aide du processus normal. Installez ensuite la nouvelle version de WebSphere eXtreme Scale Client sur votre nouvelle installation. Vous pouvez ensuite utiliser le script **xsmigration** pour déplacer les informations de configuration de WebSphere eXtreme Scale vers la nouvelle installation WebSphere Application Server.

Procédure

1. Faites migrer la configuration liée au gestionnaire de déploiement à partir de la version 7 vers la version 8.
    a. Exécutez le script de sauvegarde WebSphere Application Server. Pour plus d'informations, voir Commande WASPreUpgrade.
    b. Arrêtez le gestionnaire de déploiement.
    c. Accédez au serveur de gestionnaire de déploiement dans la configuration WebSphere eXtreme Scale Client, puis exécutez le script de migration.
        i. Accédez au répertoire `<racine_install_WXS_v8>/bin`
        ii. Exécutez la commande suivante :

```
xsmigration.bat|sh -targetwashome <WAS8x_HOME>
-sourcewashome <WAS7x_HOME> -targetprofilepath <WAS8x_DmgrProfile>
 -sourceprofilepath <WAS7x_DmgrProfile>
```

        où
        - *<WAS8x_HOME>* représente l'emplacement racine de l'installation WebSphere Application Server Version 8.x. Exemple : `/opt/IBM/WebSphere8`
        - *<WAS7x_HOME>* représente l'emplacement racine de l'installation WebSphere Application Server Version 7.x. Exemple : `/opt/IBM/WebSphere7`
        - *<WAS8x_DmgrProfile>* représente l'emplacement du profil de gestionnaire de déploiement WebSphere Application Server Version 8.x. Exemple : `/opt/IBM/WebSphere8/profiles/DMgr01`
        - *<WAS7x_DmgrProfile>* représente l'emplacement du profil de gestionnaire de déploiement WebSphere Application Server Version 7.x. Exemple : `/opt/IBM/WebSphere7/profiles/DMgr01`
2. Faites migrer la configuration liée au serveur d'applications à partir de la version 7 vers la version 8.
    a. Accédez au répertoire `<racine_install_WXS_v8>/bin`.
    b. Exécutez la commande suivante :

```
xsmigration.bat|sh -targetwashome <WAS8x_HOME>
-sourcewashome <WAS7x_HOME> -targetprofilepath <WAS8x_AppServerProfile>
 -sourceprofilepath <WAS7x_AppServerProfile>
```

    où
    - *<WAS8x_HOME>* représente l'emplacement racine de l'installation WebSphere Application

Server Version 8.x. Exemple : `/opt/IBM/WebSphere8`
- *<WAS7x_HOME>* représente l'emplacement racine de l'installation WebSphere Application Server Version 7.x. Exemple : `/opt/IBM/WebSphere7`
- *<WAS8x_AppServerProfile>* représente l'emplacement du profil de serveur d'applications WebSphere Application Server version 8.x. Exemple : `/opt/IBM/WebSphere8/profiles/AppServer01`
- *<WAS7x_AppServerProfile>* représente l'emplacement du profil de serveur d'applications WebSphere Application Server version 7.x. Exemple : `/opt/IBM/WebSphere7/profiles/AppServer01`

3. Redémarrez le gestionnaire de déploiement WebSphere Application Server version 8, puis synchronisez tous les noeuds gérés.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Mise à jour du microprogramme
Rubrique suivante : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Java

Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Vous pouvez utiliser IBM® Installation Manager pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale Client. Les groupes de correctifs peuvent être installés à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

Installation des groupes de correctifs à l'aide de l'interface graphique
Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure à l'aide de l'assistant d'IBM Installation Manager.

Installation des groupes de correctifs à l'aide de la ligne de commande
Vous pouvez utiliser IBM Installation Manager à partir de la ligne de commande pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale.

Installation des groupes de correctifs à l'aide d'un fichier de réponses
Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure avec IBM Installation Manager à l'aide d'un fichier de réponses.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Mise à jour de WebSphere eXtreme Scale Client sur WebSphere Application Server
Rubrique suivante : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Java

Installation des groupes de correctifs à l'aide de l'interface graphique

Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure à l'aide de l'assistant d'IBM® Installation Manager.

Avant de commencer

Mettez à niveau le microprogramme du dispositif avant de mettre à jour WebSphere eXtreme Scale Client. Pour plus d'informations, voir Mise à jour du microprogramme Prenez contact avec le centre de support logiciel IBM pour obtenir des informations sur les mises à niveau pour WebSphere eXtreme Scale autonome ou les offres de produit WebSphere eXtreme Scale for WebSphere Application Server. Les informations les plus récentes sont disponibles dans le centre de support logiciel IBM et dans Fix Central.

IBM Installation Manager sert à appliquer un package de maintenance de produit aux offres de produit suivantes :

- WebSphere eXtreme Scale Client dans un environnement autonome
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

Assurez-vous que l'emplacement du référentiel de service Web ou local est répertorié et vérifié ou que l'option **Search service repositories during installation and updates** est sélectionnée dans le panneau Repositories de la page des préférences Installation Manager. Pour plus d'informations sur l'utilisation des référentiels de service avec Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour mettre à niveau une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
2. Ouvrez une session sur votre système.
3. Arrêtez tous les serveurs et toutes les applications présents dans l'installation de WebSphere Application Server que vous voulez mettre à jour.
4. Démarrez Installation Manager.
5. Cliquez sur **Update**.

   Remarque : Si vous êtes invité à vous authentifier, utilisez l'ID IBM et le mot de passe vous permettant d'accéder aux sites Web protégés d'IBM.

6. Sélectionnez le groupe de packages à mettre à jour.

   Conseil : Si vous sélectionnez **Update all**, Installation Manager recherche dans tous les référentiels ajoutés et prédéfinis les mises à jour de tous les groupes de packages qu'il a installés. Utilisez cette fonction uniquement si vous contrôlez intégralement quels correctifs sont contenus dans les référentiels cible. Si vous créez et pointez vers un ensemble de référentiels personnalisés incluant uniquement les correctifs spécifiques que vous souhaitez installer, vous devez pouvoir utiliser cette fonction en toute confiance. Si vous activez la recherche dans les référentiels de service ou que vous installez les correctifs directement à partir d'autres référentiels Web en ligne, vous ne souhaitez peut-être pas sélectionner cette option afin de pouvoir sélectionner uniquement les correctifs à installer pour chaque offre dans des panneaux ultérieurs.

7. Cliquez sur **Next**.
8. Sélectionnez la version que vous voulez mettre à jour :
   - WebSphere eXtreme Scale Client dans un environnement autonome
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
   - WebSphere eXtreme Scale Client for WebSphere Application Server Version 8
9. Sélectionnez les correctifs que vous voulez installer.

   Les correctifs recommandés sont sélectionnés par défaut.

   Vous pouvez choisir de n'afficher que les correctifs recommandés et de masquer les autres.

10. Cliquez sur **Next**.
11. Acceptez les modalités du contrat de licence et cliquez sur **Next**.
12. Sélectionnez les fonctions facultatives que vous souhaitez intégrer à votre installation mise à jour.
13. Lisez le récapitulatif et cliquez sur **Update**.
    - Si l'installation a réussi, le programme affichera un message pour confirmer le succès de l'installation.
    - Si l'installation n'aboutit pas, cliquez sur **View Log File** pour corriger l'erreur.
14. Cliquez sur **Terminer**.
15. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Rubrique parent : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Désinstallation des groupes de correctifs à l'aide de l'interface graphique

Java

Installation des groupes de correctifs à l'aide de la ligne de commande

Vous pouvez utiliser IBM® Installation Manager à partir de la ligne de commande pour mettre à jour le produit à l'aide des groupes de correctifs disponibles pour les offres de produit WebSphere eXtreme Scale.

Avant de commencer

Mettez à niveau le microprogramme du dispositif avant de mettre à jour WebSphere eXtreme Scale Client. Pour plus d'informations, voir Mise à jour du microprogramme. Prenez contact avec le centre de support logiciel IBM pour obtenir des informations sur les mises à niveau pour WebSphere eXtreme Scale autonome ou les offres de produit WebSphere eXtreme Scale for WebSphere Application Server. Les informations les plus récentes sont disponibles dans le centre de support logiciel IBM et dans Fix Central.

IBM Installation Manager sert à appliquer un package de maintenance de produit aux offres de produit suivantes :

- WebSphere eXtreme Scale Client dans un environnement autonome
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 7
- WebSphere eXtreme Scale Client for WebSphere Application Server Version 8

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour mettre à niveau une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Pour obtenir la liste des groupes de correctifs et des correctifs temporaires disponibles pour WebSphere eXtreme Scale 8.5, ainsi que des informations spécifiques sur chaque correctif, procédez comme suit :
    a. Accédez à Fix Central.
    b. Sélectionnez **WebSphere** comme groupe de produits.
    c. Sélectionnez WebSphere eXtreme Scale comme produit.
    d. Sélectionnez **8.5** pour la version installée.
    e. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
    f. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
    g. Cliquez sur **Informations complémentaires** sous chaque correctif pour afficher des informations relatives au correctif.
    h. **Recommandation :** Notez le nom du groupe de correctifs que vous souhaitez installer.
2. Mettez à jour WebSphere eXtreme Scale version 8.5 à l'aide du groupe de correctifs en utilisant la procédure ci-dessous.
    - Téléchargez sur Fix Central le fichier contenant le groupe de correctifs, puis utilisez la mise à jour locale.

      Vous pouvez télécharger à partir de Fix Central un fichier compressé contenant le groupe de correctifs. Chaque fichier de groupe de correctifs compressé contient un référentiel Installation Manager pour le groupe de correctifs et comporte généralement une extension .zip. Après avoir téléchargé et extrait le fichier de groupe de correctifs, utilisez Installation Manager pour mettre à jour WebSphere Application Server version 8.x à l'aide du groupe de correctifs.

      a. Pour télécharger le groupe de correctifs, procédez comme suit :
          i. Accédez à Fix Central.
          ii. Sélectionnez **WebSphere** comme groupe de produits.
          iii. Sélectionnez **WebSphere eXtreme Scale** comme produit.
          iv. Sélectionnez **8.5** pour la version installée.
          v. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
          vi. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
          vii. Sélectionnez le groupe de correctifs à télécharger, puis cliquez sur **Continuer**.
          viii. Sélectionnez vos options de téléchargement, puis cliquez sur **Continuer**.

ix. Cliquez sur **J'accepte** pour accepter les dispositions.
x. Cliquez sur **Télécharger maintenant** pour télécharger le groupe de correctifs.
xi. Transférez le fichier compressé au format binaire vers le système sur lequel il sera installé.
xii. Extrayez les fichiers de référentiel compressés dans un répertoire de votre système.

b. Pour installer un groupe de correctifs à partir d'un fichier téléchargé, procédez comme suit :

i. Ouvrez une session sur votre système.

ii. Arrêtez tous les processus en cours d'exécution dans votre environnement. Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.

iii. Accédez au répertoire *binaires_Installation_Manager*`/eclipse/tools`, où *binaires_Installation_Manager* représente le répertoire racine d'installation d'Installation Manager.

iv. Installez le groupe de correctifs.

UNIX Linux

```
./imcl install ID_offre_version_offre,ID_fonction_facultative
  -installationDirectory emplacement_installation_produit
  -repositories emplacement_fichiers_étendus
  -acceptLicense
```

Windows

```
imcl.exe install ID_offre_version_offre,ID_fonction_facultative
  -installationDirectory emplacement_installation_produit
  -repositories emplacement_fichiers_étendus
  -acceptLicense
```

Conseils :

- La *version_offre*, qui peut, le cas échéant, être associée à l'ID offre à l'aide d'un trait de soulignement, est une version spécifique de l'offre à installer (`8.5.0.20110503_0200`, par exemple).
  - Si la *version_offre* n'est **pas** spécifiée, la version la plus récente de l'offre et **tous** les correctifs temporaires de cette version sont installés.
  - Si la *version_offre* est spécifiée, la version indiquée de l'offre est installée et **aucun** correctif temporaire de cette version n'est installé.

  La version de l'offre peut être rattachée à la fin de l'ID offre à l'aide d'un trait de soulignement lorsque vous exécutez la commande suivante au niveau du référentiel :

  ```
  imcl listAvailablePackages -repositories référentiel_source
  ```

- Vous pouvez également spécifier `none`, `recommended` ou `all` avec l'argument `-installFixes` afin d'indiquer les correctifs temporaires que vous souhaitez installer avec l'offre.
  - Si la version de l'offre n'est **pas** spécifiée, l'option `-installFixes` est paramétrée par défaut sur `all`.
  - Si la version de l'offre est spécifiée, l'option `-installFixes` est paramétrée par défaut sur `none`.
- Vous pouvez ajouter une liste de fonctions séparées par des virgules. Si aucune liste de fonctions n'est spécifiée, les fonctions par défaut sont installées.

v. **Facultatif :** Répertoriez tous les packages installés pour vérifier l'installation :

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Rubrique parent : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Désinstallation des groupes de correctifs à l'aide de la ligne de commande

Java

Installation des groupes de correctifs à l'aide d'un fichier de réponses

Vous pouvez mettre à jour WebSphere eXtreme Scale Client vers une version ultérieure avec IBM® Installation Manager à l'aide d'un fichier de réponses.

Avant de commencer

Conseil : Comme alternative à la procédure décrite dans cet article, Installation Manager vous permet d'utiliser la commande **`updateAll`** dans un fichier de réponses ou sur la ligne de commande pour rechercher et mettre à jour tous les packages installés. Utilisez cette commande uniquement si vous contrôlez intégralement quels correctifs sont contenus dans les référentiels cible. Si vous créez et pointez vers un ensemble de référentiels personnalisés incluant uniquement les correctifs spécifiques que vous souhaitez installer, vous devez pouvoir utiliser cette commande en toute confiance. Si vous activez la recherche dans les référentiels de service ou que vous installez les correctifs directement à partir d'autres référentiels Web en ligne, vous ne souhaitez peut-être pas sélectionner cette option afin de pouvoir sélectionner uniquement les correctifs à installer à l'aide de l'option -installFixes avec la commande **`install`** sur la ligne de commande ou à l'aide de l'attribut installFixes dans un fichier de réponses.

Procédure

1. Pour obtenir la liste des groupes de correctifs et des correctifs temporaires disponibles pour WebSphere eXtreme Scale Client et des informations sur chaque correctif, procédez comme suit.
    a. Accédez à Fix Central.
    b. Sélectionnez **WebSphere** comme groupe de produits.
    c. Sélectionnez WebSphere eXtreme Scale Client comme produit.
    d. Sélectionnez **8.x** pour la version installée.
    e. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
    f. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
    g. Cliquez sur **Informations complémentaires** sous chaque correctif pour afficher des informations relatives au correctif.
    h. **Recommandation :** Notez le nom du groupe de correctifs que vous souhaitez installer.
2. Mettez à jour WebSphere eXtreme Scale Client avec le correctif en procédant comme suit.
    - Téléchargez sur Fix Central le fichier contenant le groupe de correctifs, puis utilisez la mise à jour locale.

      Vous pouvez télécharger à partir de Fix Central un fichier compressé contenant le groupe de correctifs. Chaque fichier de groupe de correctifs compressé contient un référentiel Installation Manager pour le groupe de correctifs et comporte généralement une extension .zip. Après avoir téléchargé et extrait le fichier de groupe de correctifs, utilisez le gestionnaire d'installation pour mettre à jour WebSphere eXtreme Scale Client avec le groupe de correctifs.

      a. Pour télécharger le groupe de correctifs, procédez comme suit :
          i. Accédez à Fix Central.
          ii. Sélectionnez **WebSphere** comme groupe de produits.
          iii. Sélectionnez **WebSphere eXtreme Scale Client** comme produit.
          iv. Sélectionnez **8.6** comme version installée.
          v. Sélectionnez votre système d'exploitation comme plateforme, puis cliquez sur **Continuer**.
          vi. Sélectionnez **Rechercher des correctifs**, puis cliquez sur **Continuer**.
          vii. Sélectionnez le groupe de correctifs à télécharger, puis cliquez sur **Continuer**.
          viii. Sélectionnez vos options de téléchargement, puis cliquez sur **Continuer**.
          ix. Cliquez sur **J'accepte** pour accepter les dispositions.

x. Cliquez sur **Télécharger maintenant** pour télécharger le groupe de correctifs.

xi. Transférez le fichier compressé au format binaire vers le système sur lequel il sera installé.

xii. Extrayez les fichiers de référentiel compressés dans un répertoire de votre système.

b. Effectuez les actions suivantes :

i. Ouvrez une session sur votre système.

ii. Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

iii. Accédez au répertoire *`binaires_Installation_Manager`*`/eclipse/tools`, où *`binaires_Installation_Manager`* représente le répertoire racine d'installation d'Installation Manager.

iv. Installez le groupe de correctifs à l'aide d'un fichier de réponses.

Par exemple :

- Windows **Administrateur ou non administrateur :**

```
imcl.exe -acceptLicense
  input C:\temp\update_response_file.xml
  -log C:\temp\update_log.xml
  -keyring C:\IM\im.keyring
```

- UNIX Linux **Administrateur :**

```
./imcl -acceptLicense
  input /var/temp/update_response_file.xml
  -log /var/temp/update_log.xml
  -keyring /var/IM/im.keyring
```

- UNIX Linux **Non administrateur :**

```
./imcl -acceptLicense
  input rép_base_utilisateur/var/temp/update_response_file.xml
  -log rép_base_utilisateur/var/temp/update_log.xml
  -keyring rép_base_utilisateur/var/IM/im.keyring
```

Rubrique parent : Installation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Désinstallation des groupes de correctifs à l'aide de fichiers de réponses

Java

Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Vous pouvez utiliser IBM® Installation Manager pour rétrograder les offres de produit WebSphere eXtreme Scale Client vers une version précédente. Vous pouvez désinstaller les groupes de correctifs à partir de l'interface graphique ou de la ligne de commande ou à l'aide de fichiers de réponses.

Désinstallation des groupes de correctifs à l'aide de l'interface graphique
Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de l'interface graphique d'IBM Installation Manager.

Désinstallation des groupes de correctifs à l'aide de la ligne de commande
Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de la ligne de commande d'IBM Installation Manager.

Désinstallation des groupes de correctifs à l'aide de fichiers de réponses
Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente avec IBM Installation Manager à l'aide d'un fichier de réponses.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Installation des groupes de correctifs à l'aide d'IBM Installation Manager
Rubrique suivante : 2.5+ Mise à niveau de WebSphere eXtreme Scale Client for .NET

Java

Désinstallation des groupes de correctifs à l'aide de l'interface graphique

Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de l'interface graphique d'IBM® Installation Manager.

Avant de commencer

Pendant le processus de rétrogradation, Installation Manager doit accéder aux fichiers de la version précédente du package. Par défaut, ces fichiers sont enregistrés sur votre ordinateur lors de l'installation d'un package. Si vous modifiez le paramétrage par défaut ou que supprimez les fichiers enregistrés, Installation Manager a besoin d'accéder au référentiel utilisé pour l'installation de la version précédente.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour rétrograder une installation et ajouter ou supprimer une fonction.

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
    - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
2. Démarrez Installation Manager.
3. Cliquez sur **Rétrograder**.
4. Sélectionnez le groupe de packages à rétrograder.
5. Cliquez sur **Suivant**.
6. Sélectionnez la version que vous voulez rétrograder.
7. Cliquez sur **Suivant**.
8. Lisez le récapitulatif et cliquez sur **Rétrograder**.
    - Si la rétrogradation aboutit, le programme affiche un message pour confirmer le succès de la rétrogradation.
    - Dans le cas contraire, cliquez sur **View Log File** pour corriger l'erreur.
9. Cliquez sur **Terminer**.
10. Cliquez sur **Fichier > Quitter** pour fermer Installation Manager.

Rubrique parent : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Installation des groupes de correctifs à l'aide de l'interface graphique

Java

Désinstallation des groupes de correctifs à l'aide de la ligne de commande

Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente à partir de la ligne de commande d'IBM® Installation Manager.

Avant de commencer

Restriction : Pour utiliser cette procédure, Installation Manager version 1.5 ou ultérieure doit être installé sur votre système.

Pendant le processus de rétrogradation, Installation Manager doit accéder aux fichiers de la version précédente du package. Par défaut, ces fichiers sont enregistrés sur votre ordinateur lors de l'installation d'un package. Si vous modifiez le paramétrage par défaut ou que supprimez les fichiers enregistrés, Installation Manager a besoin d'accéder au référentiel utilisé pour l'installation de la version précédente.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour rétrograder une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Facultatif : Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

   Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

   Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

2. Ouvrez une session sur votre système.
3. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
4. Accédez au sous-répertoire `eclipse/tools` dans le répertoire où vous avez installé Installation Manager.
5. Utilisez la commande **`imcl`** pour rétrograder le produit.

   UNIX Linux

   ```
   ./imcl rollback ID_offre_version_offre
     -repositories référentiel_source
     -installationDirectory répertoire_installation
     -preferences clé_préférence=valeur
     -properties clé_propriété=valeur
     -keyring fichier_de_clés -password mot_de_passe
     -acceptLicense
   ```

   Windows

   ```
   imcl.exe rollback ID_offre_version_offre
   -repositories référentiel_source
   -installationDirectory répertoire_installation
   -preferences clé_préférence=valeur
   -properties clé_propriété=valeur
   -keyring fichier_de_clés -password mot_de_passe
   -acceptLicense
   ```

   Conseils :

   - La *version_offre*, qui peut, le cas échéant, être associée à l'ID offre à l'aide d'un trait de soulignement, est une version spécifique de l'offre à rétrograder (`8.5.0.20110503_0200`, par exemple).
     - Si la *version_offre* n'est **pas** spécifiée, l'installation est rétrogradée à la version précédemment installée de l'offre et **tous** les correctifs temporaires de cette version sont

installés.

- Si la *version_offre* est spécifiée, l'installation est rétrogradée à la version précédente indiquée de l'offre et **aucun** correctif temporaire de cette version n'est installé.

La version de l'offre peut être rattachée à la fin de l'ID offre à l'aide d'un trait de soulignement dans la section Package du rapport qui est généré lors de l'exécution de la commande **historyInfo** ou **genHistoryReport** à partir du répertoire *racine_serveur_app*/bin.

Pour plus d'informations sur l'utilisation d'Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

6. Facultatif : Répertoriez tous les packages installés pour vérifier la rétrogradation.

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Rubrique parent : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Installation des groupes de correctifs à l'aide de la ligne de commande

Java

Désinstallation des groupes de correctifs à l'aide de fichiers de réponses

Vous pouvez rétrograder WebSphere eXtreme Scale Client vers une version précédente avec IBM® Installation Manager à l'aide d'un fichier de réponses.

Avant de commencer

Pendant le processus de rétrogradation, Installation Manager doit accéder aux fichiers de la version précédente du package. Par défaut, ces fichiers sont enregistrés sur votre ordinateur lors de l'installation d'un package. Si vous modifiez le paramétrage par défaut ou que supprimez les fichiers enregistrés, Installation Manager a besoin d'accéder au référentiel utilisé pour l'installation de la version précédente.

Pourquoi et quand exécuter cette tâche

Restriction : Vous ne pouvez pas utiliser Installation Manager pour rétrograder une installation et ajouter ou supprimer la fonction de profil WebSphere Application Server complète .

Procédure

1. Facultatif : Si le référentiel requiert un nom d'utilisateur et un mot de passe, créez un fichier de clés pour accéder à ce référentiel.

   Pour plus d'informations sur la création d'un fichier de clés pour Installation Manager, reportez-vous au centre de documentation d'IBM Installation Manager version 1.5.

   Conseil : Lors de la création d'un fichier de clés, ajoutez `/repository.config` à la fin de l'emplacement d'URL de référentiel si la commande **`imutilsc`** ne peut pas trouver l'URL indiquée.

2. Ouvrez une session sur votre système.
3. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.
4. Utilisez un fichier de réponses pour rétrograder le produit.

   Accédez au sous-répertoire `eclipse/tools` du répertoire dans lequel vous avez installé Installation Manager, puis rétrogradez le produit.

   Par exemple :

   - Windows **Administrateur ou non administrateur :**

     ```
     imcl.exe
       input C:\temp\rollback_response_file.xml
       -log C:\temp\rollback_log.xml
       -keyring C:\IM\im.keyring
     ```

   - UNIX Linux **Administrateur :**

     ```
     ./imcl
       input /var/temp/rollback_response_file.xml
       -log /var/temp/rollback_log.xml
       -keyring /var/IM/im.keyring
     ```

   - UNIX Linux **Non administrateur :**

     ```
     ./imcl
       input rép_base_utilisateur/var/temp/rollback_response_file.xml
       -log rép_base_utilisateur/var/temp/rollback_log.xml
       -keyring rép_base_utilisateur/var/IM/im.keyring
     ```

   Remarque : Le programme peut générer des instructions post-installation importantes dans la sortie standard.

   Pour plus d'informations sur l'utilisation d'Installation Manager, accédez au centre de documentation d'IBM Installation Manager Version 1.5.

5. Facultatif : Répertoriez tous les packages installés pour vérifier la rétrogradation.

UNIX Linux

```
./imcl listInstalledPackages -long
```

Windows

```
imcl.exe listInstalledPackages -long
```

Rubrique parent : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

Tâches associées:
Installation des groupes de correctifs à l'aide d'un fichier de réponses

.NET

Mise à niveau de WebSphere eXtreme Scale Client for .NET

Pour mettre à niveau une installation existante de WebSphere eXtreme Scale Client for .NET, vous pouvez utiliser le programme d'installation. Ce dernier détecte l'installation existante et remplace les fichiers appropriés.

Avant de commencer

- Mettez d'abord à jour le microprogramme de votre dispositif avant de mettre à niveau WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Mise à jour du microprogramme.
- Téléchargez la mise à niveau de WebSphere eXtreme Scale Client for .NET. Les informations les plus récentes sont disponibles dans le centre de support logiciel IBM et dans Fix Central. Les téléchargements disponibles concernent aussi bien les nouvelles installations que les mises à niveau.

Pourquoi et quand exécuter cette tâche

La procédure d'installation remplace immédiatement votre installation existante.

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
2. Utilisez l'assistant pour installer la mise à niveau de WebSphere eXtreme Scale Client for .NET. Lorsque vous utilisez l'assistant d'installation et qu'il détecte une installation précédente, vous devez confirmer que vous voulez la mettre à niveau. Le panneau de progression de l'assistant indique les références de la version précédente et de la version de mise à niveau.

Résultats

L'ensemble du code WebSphere eXtreme Scale Client for .NET existant est remplacé sur le disque et dans le cache GAC (Global Assembly Cache). Le fichier de règles publié est installé sur le disque et dans le cache GAC, en remplacement du fichier existant.

Que faire ensuite

- Configurez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de WebSphere eXtreme Scale Client for .NET.
- Développez vos applications .NET. Pour plus d'informations, voir Développement d'applications de grille de données avec les API .NET.

.NET **Création d'une installation côte à côte de groupes de correctifs pour WebSphere eXtreme Scale Client for .NET**
Lorsque vous créez une installation côte à côte, vous pouvez utiliser plusieurs versions des groupes de correctifs pour WebSphere eXtreme Scale Client for .NET sur le même serveur. Ainsi, vos applications .NET existantes qui ont été générées avec la version précédente peuvent continuer à s'exécuter avec la version précédente du client.

Rubrique parent : Mise à jour de WebSphere DataPower XC10 Appliance
Rubrique précédente : Désinstallation des groupes de correctifs à l'aide d'IBM Installation Manager

.NET

Création d'une installation côte à côte de groupes de correctifs pour WebSphere eXtreme Scale Client for .NET

Lorsque vous créez une installation côte à côte, vous pouvez utiliser plusieurs versions des groupes de correctifs pour WebSphere eXtreme Scale Client for .NET sur le même serveur. Ainsi, vos applications .NET existantes qui ont été générées avec la version précédente peuvent continuer à s'exécuter avec la version précédente du client.

Avant de commencer

- Vous devez disposer d'un système équipé d'une version précédente de WebSphere eXtreme Scale Client for .NET, et d'un système distinct sur lequel vous installerez le groupe de correctifs plus récent.
- Installez la version plus récente de WebSphere eXtreme Scale Client for .NET sur un système distinct de vos systèmes de production. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client for .NET.

Pourquoi et quand exécuter cette tâche

Lorsque vous installez un groupe de correctifs de WebSphere eXtreme Scale Client for .NET, l'installation existante est remplacée par la nouvelle version. En fonction des caractéristiques de votre environnement, il se peut que vous préfériez effectuer des tests avant de procéder à la mise à niveau vers cette nouvelle version. Il se peut aussi que vous souhaitiez que certaines de vos applications utilisent la version précédente, et d'autres la version la plus récente. Installer manuellement WebSphere eXtreme Scale Client for .NET vous permet d'utiliser plusieurs versions en parallèle.

Lorsque vous installez une nouvelle édition importante de WebSphere eXtreme Scale Client for .NET, cette installation côte à côte a lieu automatiquement.

Procédure

1. A partir du système équipé de l'installation la plus récente, copiez l'ensemble du répertoire d'installation *net_client_home*, y compris tous ses sous-répertoires. Placez ces fichiers dans un répertoire distinct de celui contenant l'installation existante sur le système cible équipé de la version précédente. Ce répertoire est désigné sous le nom *base_côte-à-côte*.
2. Installez manuellement les assemblages WebSphere eXtreme Scale Client for .NET les plus récents dans le GAC (global assembly cache) à partir du répertoire *base_côte-à-côte*/bin. Après cette installation, le GAC contient l'ancienne et la nouvelle version des assemblages WebSphere eXtreme Scale Client for .NET. La procédure d'installation manuelle des assemblages dans le GAC dépend de la version de Windows et de l'infrastructure .NET installée sur votre système. Par exemple, vous pouvez utiliser l'explorateur Windows pour copier les fichiers des assemblages dans le répertoire %systemroot%\assembly. Vous pouvez également utiliser gacutil.exe, un utilitaire téléchargeable à partir du site Web de Microsoft.
3. Mettez à jour vos applications WebSphere eXtreme Scale Client for .NET pour qu'elles utilisent la nouvelle version. Utilisez l'une des options suivantes :
   - Ajoutez un élément de redirection d'assemblage au fichier de configuration de chaque application à mettre à jour. Cet élément redirige toute référence à l'ancien assemblage WebSphere eXtreme Scale Client for .NET vers le nouvel assemblage.

     ```
     <runtime>
       <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
          <dependentAssembly>
             <assemblyIdentity name="IBM.WebSphere.Caching"
     publicKeyToken="b439a24ee43b0816" />
             <bindingRedirect oldVersion="8.6.0.0-8.6.0.1" newVersion="8.6.0.2" />

          </dependentAssembly>
       </assemblyBinding>
     </runtime>
     ```

   - Régénérez votre application WebSphere eXtreme Scale Client for .NET en incluant le nouveau fichier d'assemblage dans la liste des références d'assemblage pour votre projet d'application.

Résultats

- Les applications ainsi mises à jour utiliseront la nouvelle version de WebSphere eXtreme Scale Client for .NET, tandis que les autres continueront d'utiliser la version précédente.
- Les fichiers journaux des applications qui utilisent la nouvelle version se trouvent dans le même

répertoire que les fichiers journaux des applications qui utilisent l'ancienne version.
- La configuration du client par défaut est toujours obtenue à partir de l'ancienne installation de WebSphere eXtreme Scale Client for .NET (répertoire *net_client_home*/config). Pour utiliser un autre fichier de propriétés, transmettez explicitement son chemin d'accès à l'API Connect().

Avertissement : Vous devez manuellement désinstaller vos modifications. Les installations manuelles ne sont pas mises à niveau par le programme d'installation de WebSphere eXtreme Scale Client for .NET.

Rubrique parent : 2.5+ Mise à niveau de WebSphere eXtreme Scale Client for .NET

Configuration de votre dispositif

Après avoir initialisé le dispositif à l'aide de la console série, vous devez le configurer à l'aide de l'interface utilisateur pour activer ses fonctionnalités.

Avant de commencer

- Installez et initialisez le matériel du dispositif. Pour plus d'informations, voir Installation de WebSphere DataPower XC10 Appliance.
- Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Le processus d'initialisation à l'aide de la console série prépare le dispositif pour son administration à partir de l'interface utilisateur. Pour préparer le dispositif pour son utilisation, vous devez configurer des paramètres supplémentaires.

Remarque : Tous ces paramètres sont importants, mais l'ajout correct d'un serveur DNS (Domain Name System), d'un serveur NTP (Network Time Protocol) et la configuration de la distribution du courrier revêt une importance toute particulière. Ces tâches doivent être effectuées avant toute autre utilisation.

Sécurité
Vous pouvez configurer plusieurs éléments de sécurité dans le dispositif, notamment la sécurité de l'interface utilisateur et du transport.

Gestion des utilisateurs et des groupes
Des utilisateurs et des groupes sont fournis pour gérer les accès à votre dispositif WebSphere DataPower XC10 Appliance par chaque individu. Vous pouvez utiliser des groupes d'utilisateurs pour appliquer des autorisations à des groupes d'utilisateurs.

Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Lors de la connexion série initiale, vous avez configuré l'interface Ethernet *mgmt* afin de connecter le dispositif à votre réseau. Vous pouvez définir des ports Ethernet privés supplémentaires dans l'interface utilisateur.

Importation et exportation de configurations
Lorsque vous configurez un nouveau dispositif, vous pouvez exporter et stocker les paramètres de configuration pour ce dispositif. Si ultérieurement des modifications sont effectuées qui nécessitent la suppression ou la réinstallation du dispositif, vous pouvez alors importer les données de configuration stockées sans perdre les paramètres de configuration.

Gestion du serveur DNS (Domain Name System)
Un serveur DNS (Domain Name System) est requis pour le système IBM® WebSphere DataPower XC10 Appliance étant donné que les services de recherche DNS sont utilisés pour la communication. Vous devez spécifier ce serveur DNS lors de l'initialisation du dispositif.

Mappage d'adresses IP vers des noms d'hôte
Avant que les informations relatives à une adresse puissent être utilisées pour créer une connexion dans un protocole de réseau TCP/IP, l'adresse IP doit être associée à un nom d'hôte. Vous pouvez résoudre une adresse IP en nom d'hôte en éditant le fichier etc/hosts du dispositif. Vous pouvez éditer le fichier etc/hosts à partir de l'interface utilisateur.

Gestion des paramètres de date et heure
Utilisez des serveurs NTP (Network Time Protocol) pour maintenir la synchronisation de la date et de l'heure entre les différents dispositifs de votre collectivité.

Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
La fonction de distribution du courrier du dispositif est utilisée afin de redéfinir les mots de passe des utilisateurs. Lorsqu'un utiliseur demande un nouveau mot de passe, il le reçoit dans un courrier électronique généré par le dispositif.

Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Le système WebSphere DataPower XC10 Appliance peut être redémarré ou arrêté à partir de l'interface utilisateur.

Sécurité

Vous pouvez configurer plusieurs éléments de sécurité dans le dispositif, notamment la sécurité de l'interface utilisateur et du transport.

IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
IBM WebSphere DataPower XC10 Appliance permet de contrôler l'accès au dispositif et aux données grille de données contenues dans le dispositif.

Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Une grande partie des fonctions de sécurité offertes par WebSphere DataPower XC10 Appliance sont déjà intégrées dans le dispositif lors de sa fabrication. D'autres paramètres sont inclus pour proposer des options de sécurité supplémentaires pour votre environnement.

Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Si vous le souhaitez, vous pouvez utiliser un annuaire LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs sur votre système IBM WebSphere DataPower XC10 Appliance.

Gestion des utilisateurs et des groupes
Des utilisateurs et des groupes sont fournis pour gérer les accès à votre dispositif WebSphere DataPower XC10 Appliance par chaque individu. Vous pouvez utiliser des groupes d'utilisateurs pour appliquer des autorisations à des groupes d'utilisateurs.

Activation de la sécurité pour les grilles de données
Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

Java **Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.**
Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Java **Configuration d'une application de grille de données pour l'utilisation de l'authentification client**
Si la grille de données que vous configurez pour l'application utilise la sécurité, vous devez configurer un fichier `client.properties` comportant des paramètres à transmettre à l'application de grille de données.

Configuration de TLS pour les applications de grille de données
Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Passerelle REST : Configuration de la sécurité
Pour accéder à une grille de données via la passerelle REST, l'utilisateur doit être authentifié sur WebSphere DataPower XC10 Appliance, que la sécurité de la grille de données ait été activée ou non. Le client d'application doit toujours fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP. Pour accéder à des grilles de données via la passerelle REST, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

IBM® WebSphere DataPower XC10 Appliance permet de contrôler l'accès au dispositif et aux données grille de données contenues dans le dispositif.

Sécurité du dispositif

Plusieurs caractéristiques critiques contribuent à sécuriser le dispositif, notamment :

Le dispositif est implanté dans un boîtier résistant aux intrusions

Un commutateur de détection d'intrusion présent sur le boîtier est sous surveillance permanente. Si le commutateur est déclenché, le dispositif ne démarre plus. Le dispositif doit alors être envoyé à IBM pour permettre son redémarrage. Des éléments supplémentaires, comme des vis anti-intrusion, sont également inclus pour décourager l'ouverture du boîtier. La conception du dispositif permet d'accéder aux éléments remplaçables par l'utilisateur depuis l'arrière du dispositif sans ouvrir le boîtier.

Aucun accès au système d'exploitation n'est possible via un interpréteur de commandes.

Le système d'exploitation du dispositif ne comporte pas d'interpréteur de commandes. L'absence d'interpréteur de commandes a été conçue délibérément pour réduire le périmètre de vulnérabilité du dispositif. Un seul ID utilisateur du système d'exploitation est utilisé sur le dispositif. Etant donné l'absence d'interpréteur de commandes, il n'est pas possible de se connecter depuis l'extérieur du dispositif avec un ID utilisateur.

Pas de possibilité d'exécution de logique soumise par l'utilisateur sur le dispositif

Le dispositif ne fournit pas la possibilité à un utilisateur de télécharger un script ou un code exécutable. L'unique exception à cette règle concerne une mise à jour du microprogramme du système au cours de laquelle vous exécutez un script pour installer le microprogramme mis à jour dans le dispositif. Ces mises à jour comportent, par précaution, une signature de l'éditeur du microprogramme. Aucun logiciel non accrédité soumis par l'utilisateur ne peut être exécuté sur le dispositif.

Grille de données : Sécurité

Vous pouvez gérer l'accès aux informations contenues dans vos grilles de données. Si vous n'activez pas la sécurité sur votre grille de données, les informations des grille de données sont accessibles par toutes les applications. Vous pouvez activer la sécurité générale sur une grille de données, afin d'autoriser tous les utilisateurs possédant un compte et un mot de passe sur le dispositif à accéder à la grille de données. Vous pouvez également restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs en activant des autorisations pour la grille de données.

TLS (Transport Layer Security)

Vous pouvez utiliser TLS pour protéger les grilles de données et l'interface utilisateur en configurant un fichier de clés, un fichier de clés certifiées et des alias de certificat. Les paramètres TLS s'appliquent à tous les dispositifs de la collectivité.

Utilisateurs et groupes d'utilisateurs

Vous pouvez définir des permissions pour les utilisateurs et les groupes d'utilisateurs. Ces permissions peuvent s'appliquer à l'administration du dispositif et à la sécurité de la grille de données.

Rubrique parent : Sécurité

Concepts associés:
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration de la sécurité du client

Référence associée:
Passerelle REST : Configuration de la sécurité
Fichier de propriétés du client

Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Avant de commencer

- Vous pouvez configurer TLS avec la version 1.0.0.4 ou une version suivante.
- Vous devez utiliser le correctif WebSphere eXtreme Scale Client version 7.1 ou suivante.
- Vous devez disposer de droits d'accès d'administrateur du dispositif.
- Vous devez disposer d'un fichier de clés ou d'un fichier de clés certifiées avec les mots de passe associés à ajouter à la configuration du dispositif.
- Pour modifier le fichier de clés certifiées existant, vous pouvez le télécharger à partir du dispositif.
- Vous devez mettre à jour le fichier de clés certifiées à l'aide des certificats publics des clients. Le dispositif doit accréditer les clients qui se connectent.
- Le fichier de clés certifiées fourni doit inclure un certificat public correspondant à une entrée dans le fichier de clés. Les alias de certificat issus du fichier de clés doivent être accrédités dans le fichier de clés certifiées à fournir sous la forme d'une option de configuration possible pour l'alias de certificat du dispositif.
- Le paramètre de sécurité globale WebSphere Application Server détermine comment le serveur tente de se connecter à WebSphere DataPower XC10 Appliance :
  - Lorsque le paramètre de sécurité globale est désactivé, des tentatives de connexion sur TCP/IP sont effectuées.
  - Lorsque le paramètre de sécurité globale est activé, vous devez ajouter le certificat public du dispositif aux fichiers de clés certifiées de WebSphere Application Server.

  Si le paramètre **TLS requis** de votre WebSphere DataPower XC10 Appliance est configuré, vous devez activer la sécurité globale. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Les paramètres TLS s'appliquent à l'interface utilisateur et aux grilles de données. Les paramètres sont appliqués à tous les dispositifs de la collectivité.

Procédure

1. **Requise pour WebSphere Application Server :** Ajoutez le certificat public du dispositif aux fichiers de clés certifiées par défaut de WebSphere Application Server.
   - **Si vous utilisez le fichier de clés certifiées par défaut du dispositif :**

     Exécutez le script **addXC10PublicCert.py** à partir du répertoire *racine_was*/bin du gestionnaire de déploiement. Utilisez la syntaxe suivante :

     ```
     wsadmin -lang jython -f addXC10PublicCert.py
     ```

   - **Si vous utilisez des clés personnalisées pour le dispositif :**

     Exécutez le script **addXC10PublicCert.py** à partir du répertoire *racine_was*/bin du gestionnaire de déploiement, à l'aide de l'option de ligne de commande **-certPath**. La valeur de l'option de ligne de commande **-certPath** est l'emplacement disque du certificat public qui correspond à l'alias configuré pour le fichier de clés certifiées dans le dispositif.

     ```
     wsadmin -lang jython -f addXC10PublicCert.py -certPath ./trustStore.jks
     ```

2. **Requise pour WebSphere Application Server :** Téléchargez le fichier de clés certifiées du dispositif et les certificats publics de WebSphere Application Server, puis exécutez l'utilitaire keytool pour ajouter le certificat dans le fichier de clés certifiées. Cet outil met à jour le fichier de clés certifiées du dispositif pour inclure les certificats émanant de WebSphere Application Server.
   a. Si vous utilisez le fichier de clés certifiées par défaut du dispositif, téléchargez le fichier de clés certifiées actif. Cliquez sur **Collectivité** > **Paramètres** > **Transport Layer Security (TLS)**. Cliquez sur **Télécharger un fichier de clés certifiées actif** et rappelez-vous l'emplacement dans lequel vous avez enregistré le fichier sur disque, par exemple dans le répertoire `/downloads/trustStore.jks`.
   b. Extrayez le certificat public de WebSphere Application Server.
      i. Dans la console d'administration de WebSphere Application Server, cliquez sur **Sécurité** > **Certificat SSL et gestion des clés** > **Fichiers de clés et certificats**.

ii. Dans **Utilisations des fichiers de clés**, sélectionnez **Fichier de clés des certificats racine**.
iii. Sélectionnez **DmgrDefaultRootStore**.
iv. Sélectionnez **Certificats personnels**.
v. Cliquez sur la case à cocher en regard d'un certificat dans le fichier de clés root. Indiquez le nom de fichier qualifié complet du certificat à extraire, par exemple : `/certificates/public.cer`.
vi. Dans une fenêtre de ligne de commande, exécutez la commande suivante : cd `/java_home/bin`
vii. Exécutez l'utilitaire keytool.

```
keytool -import -noprompt -alias "example alias" -keystore
/downloads/trustStore.jks  -file /certificates/public.cer -storepass
xc10pass -storetype jks
```

viii. Si vous avez d'autres certificats à importer, répétez les étapes d'extraction des certificats, puis relancez l'utilitaire keytool.

3. Téléchargez les informations de fichier de clés et de fichier de clés certifiées vers le dispositif. Dans l'interface utilisateur du dispositif, cliquez sur **Collectivité** > **Paramètres** > **Transport Layer Security (TLS)**. Si vous avez terminé les étapes de WebSphere Application Server, téléchargez le fichier `/downloads/trustStore.jks` mis à jour. Vous devez ensuite mettre à jour le mot de passe associé. Si vous utilisez le fichier de clés certifiées par défaut, le mot de passe est `xc10pass`.
4. Sélectionnez l'alias de certificat de la collectivité.
5. Définissez le type de transport. Choisissez l'un des paramètres de type de transport suivants :
   - **TLS pris en charge** : Les grilles de données communiquent à l'aide de TCP/IP, SSL ou TLS. L'interface utilisateur est accessible à l'aide de HTTP et HTTPS.
   - **TLS requis** : Les grilles de données communiquent à l'aide de SSL ou TLS uniquement. L'interface utilisateur est accessible à l'aide de HTTPS uniquement.
   - : Les grilles de données communiquent à l'aide de connexions non sécurisées. L'interface utilisateur est accessible à l'aide de HTTP et HTTPS.
6. Pour que le client envoie un certificat accrédité pour activer la communication, sélectionnez **Activer l'authentification du certificat client**.
7. Cliquez sur **Soumettre les paramètres TLS** pour enregistrer les modifications de la configuration.

Résultats

La collectivité doit redémarrer pour terminer l'application des modifications à la configuration TLS.

Certaines parties de l'interface utilisateur sont accessibles lorsque la collectivité redémarre. Si vous ne pouvez pas accéder à des parties de l'interface utilisateur, attendez un certain temps et renvoyez la demande. Le panneau des tâches indique certaines modifications automatiquement en affichant l'état d'aboutissement.

Si vous avez modifié l'alias de certificat utilisé par le dispositif, il peut être nécessaire de redémarrer le navigateur, de vous déconnecter et de vous reconnecter à l'interface utilisateur ou d'accréditer de nouveaux certificats à partir d'une invite de navigateur.

Si l'interface utilisateur semble indisponible lorsque l'authentification du client est activée, vérifiez que vous avez importé un certificat de client accrédité vers le navigateur. Si vous ne l'avez pas fait, vous ne pouvez pas accéder à l'interface utilisateur. Une fois connecté à l'interface utilisateur, la tâche indique l'aboutissement de la configuration TLS.

Que faire ensuite

Meilleures pratiques

- Pour éviter les avertissements de navigateur lorsque vous accédez à l'interface utilisateur à partir de dispositifs différents, incluez un caractère générique dans le nom usuel du certificat dans le fichier de clés certifiées. Chaque dispositif utilise le même certificat pour la configuration TLS que celle définie par l'alias de certificat. Par exemple, vous pouvez utiliser `*.mycompany.com` au lieu de `myhost.mycompany.com` pour que le certificat soit valide pour tous les hôtes du domaine `mycompany`.
- Vous pouvez utiliser une autorité de certificat (CA) privée pour signer le certificat associé à l'alias de certificat que vous avez choisi pour la configuration TLS. Ensuite, vous pouvez importer le certificat CA vers le navigateur et accréditer n'importe quelle collectivité avec un certificat signé par la CA privée sans afficher d'invite. L'utilisation d'une CA privée s'applique généralement uniquement pour l'accès à un intranet privé.

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Surveillance avec l'utilitaire xscmd

Référence associée:
Passerelle REST : Configuration de la sécurité
Fichier splicer.properties

Information associée:
Gestion des clés avec l'interface graphique IKEYMAN

Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance

Une grande partie des fonctions de sécurité offertes par WebSphere DataPower XC10 Appliance sont déjà intégrées dans le dispositif lors de sa fabrication. D'autres paramètres sont inclus pour proposer des options de sécurité supplémentaires pour votre environnement.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pour vous familiariser avec les fonctions de sécurité intégrées dans le dispositif, consultez la rubrique IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité.

Pourquoi et quand exécuter cette tâche

En supplément des fonctions de sécurité du dispositif, vous pouvez configurer plusieurs options disponibles pour contrôler le comportement de l'utilisateur.

Procédure

1. Accédez au panneau Paramètres. Pour gérer vos options de sécurité, accédez au panneau Paramètres à l'aide d'une des méthodes suivantes :
   - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
   - Dans la page Bienvenue, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Sécurité**.
3. Définissez vos autorisations de sécurité.
   a. Configurez la zone **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. La valeur par défaut de cette zone est *Désactivé*. Elle permet de spécifier si l'utilisateur est autorisé à créer son propre compte. Lorsque la valeur Activée est sélectionnée, un bouton **S'enregistrer** apparaît sur l'écran de connexion. Pour plus d'informations sur l'auto-enregistrement de l'utilisateur, reportez-vous à la rubrique Enregistrement d'un nouveau compte utilisateur.
   b. Configurez la zone *Permettre la réinitialisation du mot de passe à partir de la console série*. La valeur par défaut de cette zone est *Désactivé*.

      Désactivé : veillez à configurer un serveur SMTP et une adresse électronique pour l'utilisateur xcadmin. Ces configurations permettent de réinitialiser le mot de passe dans le cas où l'utilisateur xcadmin viendrait à perdre son mot de passe. Si cette zone est désactivée et que ces configurations ne sont pas effectuées, il est impossible de réinitialiser le mot de passe xcadmin. Dans ce cas, le dispositif devra être retourné à IBM pour réinitialisation.

      Activé : Vous pouvez réinitialiser le mot de passe de l'utilisateur xcadminà l'aide d'une connexion série sans autres données d'identification ni message SMTP. Si cette option est sélectionnée, le contrôle de l'accès physique à votre système WebSphere DataPower XC10 Appliance est encore renforcé par rapport à la normale. En effet, par le biais d'un accès physique à la machine, n'importe quel utilisateur peut obtenir un accès en tant qu'administrateur du dispositif.
4. Configurez votre dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP (Lightweight Directory Access Protocol). Pour plus d'informations sur la configuration du dispositif afin d'authentifier les utilisateurs avec un annuaire LDAP, voir Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP.

Résultats

A l'issue de ces étapes, vous aurez déterminé comment le dispositif gère certains scénarios affectant la sécurité et spécifié si une authentification externe est utilisée pour permettre l'accès à l'interface utilisateur.

Que faire ensuite

Configurez des utilisateurs et des groupes pour fournir l'accès à l'interface utilisateur. Vous utilisez également des utilisateurs et des groupes pour fournir l'accès aux grilles de données.

Rubrique parent : Sécurité

Concepts associés:

IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données

Référence associée:
Passerelle REST : Configuration de la sécurité

Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Si vous le souhaitez, vous pouvez utiliser un annuaire LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs sur votre système IBM® WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces opérations.

Pourquoi et quand exécuter cette tâche

L'utilisation d'un serveur LDAP pour authentifier les utilisateurs est facultative. L'attribut *nom d'utilisateur* est utilisé pour authentifier les utilisateurs d'IBM WebSphere DataPower XC10 Appliance dans l'annuaire LDAP. Les utilisateurs ne figurant pas dans l'annuaire LDAP ne peuvent pas être authentifiés.

2.5+ Vous pouvez configurer WebSphere DataPower XC10 Appliance pour qu'il procède à l'authentification LDAP via une connexion SSL. Pour ce faire, vous devez spécifier une URL LDAPS (Lightweight Directory Access Protocol over SSL) pour la connexion au serveur LDAP. Si vous utilisez LDAPS, le fichier de clés certifiées du dispositif doit être modifié pour qu'il accepte le certificat SSL du serveur LDAP. Si ce certificat a été émis par une autorité de certification, le certificat de signataire racine de cette autorité doit être ajouté au fichier de clés certifiées du dispositif. Si le certificat SSL du serveur LDAP est autosigné, il doit être ajouté au fichier de clés certifiées du dispositif.

2.5+ Lorsque l'authentification LDAP est configurée, tout ID utilisateur LDAP qui correspond au nom de domaine configuré de base et au filtre de recherche configuré pour les utilisateurs peut s'authentifier sur le dispositif. Cet utilisateur possède les droits et l'accès à la grille de données qui lui sont accordés ainsi que les droits d'accès et l'accès à la grille de données qui sont accordés aux groupes LDAP auxquels il appartient. Il n'est pas nécessaire d'ajouter l'utilisateur à la collectivité. Tous les utilisateurs LDAP ont également les droits et l'accès qui sont accordés au groupe Tout le monde. Vous pouvez ajouter un utilisateur à la collectivité afin que l'accès à la grille de données et les droits puissent être configurés pour cet utilisateur.

Lorsque l'authentification LDAP est configurée, vous ne pouvez ajouter que des groupes LDAP à la collectivité. L'accès et les droits peuvent être accordés aux groupes spécifiés.

Lorsque l'authentification LDAP est configurée, vous ne pouvez pas utiliser la console d'administration pour le dispositif ou des interfaces de programmation du dispositif pour ajouter ou supprimer des membres d'un groupe. L'appartenance à un groupe doit être gérée avec vos outils d'administration d'annuaire LDAP.

Pour les versions de IBM WebSphere DataPower XC10 Appliance antérieures à la version 2.5, seuls les utilisateurs qui sont spécifiquement ajoutés à la collectivité bénéficient de droits et d'accès. La prise en charge d'un LDAP généralisé est ajouté en version 2.5. Pour les versions antérieures à la version 2.5, le dispositif importe les appartenances de groupe de LDAP lorsque le groupe est ajouté à la collectivité. Le groupe est ensuite géré sur le dispositif, et l'appartenance des membres peut diverger de ce qui a été stocké dans LDAP. A partir de la version 2.5, si l'authentification LDAP est configurée, les appartenances à un groupe sont toujours résolues par l'interrogation du serveur LDAP.

Remarques sur la migration : Dans une collectivité qui inclut la version 2.5, ainsi que les dispositifs qui exécutent des versions de microprogramme antérieures, les utilisateurs qui sont dans LDAP sans être stockés dans la collectivité du dispositif ne peuvent pas accéder à des ressources restreintes sur des périphériques avec l'ancien microprogramme. Par conséquent, les clients peuvent uniquement utiliser les ID utilisateur qui sont ajoutés à la collectivité du dispositif jusqu'à ce que tous les périphériques soient mis à niveau.

Lorsqu'une collectivité comprend des membres qui exécutent des versions de microprogramme antérieures à la version 2.5, il est possible que les appartenances de groupe qui sont stockées sur d'anciens dispositifs divergent de ce qui est stocké dans LDAP. Ces incohérences peuvent provoquer des problèmes. Par exemple, si un ID utilisateur est dans un groupe qui est stocké sur un dispositif plus ancien et si les autorisations et les accès sont associés à ce groupe mais que ce groupe n'existe pas dans LDAP, ou si l'appartenance au groupe qui est stockée sur le dispositif diffère de ce qui figure dans LDAP, l'ID utilisateur ne pourra peut-être pas accéder à des ressources restreintes sur le nouveau dispositif. Ce comportement est dû au fait que le dispositif en version 2.5 vérifie LDAP directement, et non pas n'importe quelle version locale de l'appartenance à un groupe. Lorsque vous effectuez une migration, vérifiez que les ID utilisateur qui sont utilisés pour accéder aux données du dispositif disposent des droits et des accès nécessaires. Vérifiez également que ces ID utilisateur représentent des membres des groupes LDAP qui disposent des autorisations requises.

Procédure

1. Accédez au panneau **Paramètres**. Procédez de l'une des manières suivantes :

- Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
- Dans la page Bienvenue, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.

2. Développez l'entrée **Sécurité**.
3. Configurez votre dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP.

   a. Pour activer ce mécanisme, cochez la case en regard de l'option **Activer l'authentification LDAP**. Par défaut, la case **Activer l'authentification LDAP** n'est pas sélectionnée. La sélection de cette case indique à WebSphere DataPower XC10 Appliance d'utiliser le serveur LDAP pour authentifier les utilisateurs lors de leur connexion.

   b. Entrez l'URL du fournisseur JNDI. Exemple pour un annuaire LDAP non SSL :

      ```
      ldap://mycompany.com:389/
      ```

      ou

      ```
      ldap://mycompany.com/
      ```

      Si aucun port n'est spécifié de manière explicite, le numéro de port par défaut est 389. Exemple pour un annuaire LDAP SSL :

      ```
      ldaps://mycompany.com:636/
      ```

      ou

      ```
      ldaps://mycompany.com/
      ```

      Si aucun port n'est spécifié de manière explicite, le numéro de port par défaut est 636.

   c. Entrez le nom distinctif JNDI de base (utilisateurs). Exemple :

      ```
      CN=users,DC=mycompany,DC=com
      ```

   d. Entrez le nom distinctif JNDI de base (groupes). Exemple :

      ```
      DC=mycompany,DC=com
      ```

   e. Entrez le filtre de recherche (utilisateurs). Exemple :

      ```
      (&(sAMAccountName={0})(objectcategory=user)) ou uid={0}
      ```

      Remarque : L'ID utilisateur est incorporé dans l'espace réservé "{0}". "{0}" est remplacé par l'ID utilisateur de connexion saisi dans l'écran de connexion.

   f. Entrez l'authentification de sécurité JNDI. Cette zone est facultative sauf si votre serveur LDAP n'autorise pas les requêtes LDAP anonymes. Exemple :

      ```
      CN=Administrator,CN=users,DC=mycompany,DC=com
      ```

   g. Entrez le mot de passe. Cette zone correspondant aux données d'identification de sécurité JNDI est facultative, sauf si votre serveur LDAP n'autorise pas les requêtes LDAP anonymes.

4. Testez les paramètres d'authentification LDAP que vous avez définis. Vous pouvez tester les paramètres que vous avez définis pour configurer l'authentification avec le serveur LDAP. Cette section vous permet d'exécuter des requêtes LDAP afin de rechercher des utilisateurs ou des groupes.

   a. Cliquez sur **Test des paramètres d'authentification LDAP** pour développer cette section.

   b. Pour tester un nom d'utilisateur, entrez un nom d'utilisateur dans la zone Utilisateur LDAP, puis cliquez sur le bouton **Test de requête LDAP**. Exemple :

      ```
      utilisateur_test@us.ibm.com
      ```

      Si la requête aboutit, le message suivant s'affiche : *Nom distinctif d'utilisateur LDAP trouvé : <informations utilisateur>*. Si la requête échoue, un message d'erreur s'affiche.

   c. Pour tester un nom de groupe, entrez un nom de groupe dans la zone Groupe LDAP, puis cliquez sur le bouton **Test de requête LDAP** associé. Exemple :

```
Groupe test
```

Si la requête aboutit, le message suivant s'affiche : *Nom distinctif de groupe LDAP trouvé : <informations utilisateur>*. Si la requête échoue, un message d'erreur s'affiche.

Résultats

Vous avez défini un annuaire LDAP pour authentifier les utilisateurs externes qui accèdent à l'interface utilisateur.

Que faire ensuite

La maîtrise du contrôle d'accès des utilisateurs aux différentes zones de votre environnement constitue un élément important de votre solution de sécurité. Pour plus d'informations sur la gestion des utilisateurs et des groupes et de leurs autorisations, reportez-vous à la rubrique Gestion des utilisateurs et des groupes.

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données

Référence associée:
Passerelle REST : Configuration de la sécurité

Gestion des utilisateurs et des groupes

Des utilisateurs et des groupes sont fournis pour gérer les accès à votre dispositif WebSphere DataPower XC10 Appliance par chaque individu. Vous pouvez utiliser des groupes d'utilisateurs pour appliquer des autorisations à des groupes d'utilisateurs.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Vous pouvez gérer vos utilisateurs et vos groupes d'utilisateurs à partir de l'interface utilisateur de WebSphere DataPower XC10 Appliance .

Création d'un utilisateur
Un nom d'utilisateur et un mot de passe sont requis pour se connecter à l'interface utilisateur. Procédez comme suit pour créer de nouveaux comptes permettant à leurs utilisateurs d'accéder et de gérer votre WebSphere DataPower XC10 Appliance.

Gestion des utilisateurs
Une fois un utilisateur créé, vous devez le modifier manuellement dans le cas où d'autres autorisations sont nécessaires. Vous pouvez également suivre ces étapes pour modifier un utilisateur si l'information a changé.

Suppression d'un utilisateur
Un nom d'utilisateur et un mot de passe sont requis pour se connecter à interface utilisateur. Lorsque vous n'avez plus besoin d'un utilisateur, vous pouvez le supprimer de WebSphere DataPower XC10 Appliance.

Enregistrement d'un nouveau compte utilisateur
Vous pouvez créer votre propre compte utilisateur lorsque l'administrateur active l'option **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. Utilisez cette tâche pour créer un compte utilisateur à partir de l'écran de connexion.

Création d'un groupe d'utilisateurs
Vous pouvez créer des groupes d'utilisateurs afin de mieux gérer l'accès de vos utilisateurs à des ressources WebSphere DataPower XC10 Appliance spécifiques.

Gestion des groupes d'utilisateurs
Lorsque vous créez un groupe d'utilisateurs, ce dernier ne dispose pas de membres. Vous devez ajouter manuellement les utilisateurs au groupe si l'authentification LDAP n'est pas activée.

Suppression d'un groupe d'utilisateurs
Vous pouvez supprimer un groupe d'utilisateurs de WebSphere DataPower XC10 Appliance si le groupe d'utilisateurs est devenu inutile.

Droits utilisateur
Les droits utilisateur permettent de déterminer quels panneaux sont visibles par chaque utilisateur et les autorisations d'accès dont il dispose pour un objet spécifique.

Rubrique parent : Configuration de votre dispositif
Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Sécurité
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Création d'un utilisateur

Un nom d'utilisateur et un mot de passe sont requis pour se connecter à l'interface utilisateur. Procédez comme suit pour créer de nouveaux comptes permettant à leurs utilisateurs d'accéder et de gérer votre WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces opérations. Si vous utilisez LDAP (Lightweight Directory Access Protocol) pour authentifier les utilisateurs, l'utilisateur en cours d'enregistrement doit préalablement exister dans le référentiel LDAP. Pour plus d'informations sur la création d'une configuration LDAP, reportez-vous à la rubrique Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP .

Pourquoi et quand exécuter cette tâche

Procédez comme suit pour créer un utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance :

Procédure

1. Accédez au panneau **Utilisateurs**.
   - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance , accédez à **Collectivité** > **Utilisateurs**.
   - Dans le panneau d'**accueil**, cliquez sur le lien **Créer des utilisateurs** dans la section **Etape 1 : configurez l'appareil**.
2. Cliquez sur l'icône Ajouter ( ) pour lancer l'ajout d'un nouvel utilisateur.
3. Entrez un ID dans la zone **Nom d'utilisateur**. La valeur de cette zone peut comporter jusqu'à 64 caractères et ne doit pas être vide. Vous pouvez utiliser tous les caractères alphanumériques et la plupart des caractères spéciaux. Vous ne pouvez pas utiliser d'espaces ni les caractères spéciaux suivants : < #. Cette zone ne peut plus être modifiée après la création de l'utilisateur. Si vous utilisez LDAP, l'utilisateur en cours d'enregistrement doit préalablement exister dans le référentiel LDAP.
4. Facultatif : Entrez le nom de l'utilisateur dans la zone **Nom complet**. Cette zone est utilisée à des fins d'affichage dans l'interface utilisateur. Si vous n'entrez pas de valeur dans cette zone, le nom d'utilisateur s'affiche. Une fois l'utilisateur créé, seul l'utilisateur peut modifier la zone. L'administrateur ne peut pas changer la valeur de la zone après la création de l'utilisateur.
5. Entrez le mot de passe de l'utilisateur dans la zone **Mot de passe**. Le mot de passe peut contenir les mêmes caractères disponibles pour la zone **Nom d'utilisateur**. Si le protocole SMTP (Simple Mail Transfer Protocol) est activé, vous pouvez ne pas renseigner la zone du mot de passe lors de la création d'un utilisateur ; un mot de passe est créé automatiquement. Si l'authentification LDAP est activée, la zone **Mot de passe** n'est pas affichée puisque le mot de passe de l'annuaire LDAP est utilisé pour l'authentification. Entrez le même mot de passe de l'utilisateur dans la zone **Vérification du mot de passe**.
6. Entrez une adresse électronique valide pour l'utilisateur dans la zone **Adresse électronique**. Cette zone spécifie l'adresse électronique utilisée pour fournir un nouveau mot de passe si l'utilisateur a oublié sont mot de passe et d'autres notifications. L'adresse électronique est nécessaire lorsqu'un serveur SMTP (Simple Mail Transfer Protocol) est utilisé. Voir Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance pour plus d'informations.
7. Cliquez sur **OK**.

Résultats

Vous disposez d'un nouveau compte utilisateur que vous pouvez utiliser pour vous connecter à l'interface utilisateur.

Que faire ensuite

A sa création initiale, l'utilisateur ne dispose que des autorisations par défaut. Si vous désirez lui accorder des autorisations supplémentaires, consultez la rubrique Gestion des utilisateurs pour les instructions correspondantes. Pour ajouter le nouvel utilisateur à un groupe d'utilisateurs, reportez-vous à la rubrique Gestion des groupes d'utilisateurs.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:

Gestion des utilisateurs

Une fois un utilisateur créé, vous devez le modifier manuellement dans le cas où d'autres autorisations sont nécessaires. Vous pouvez également suivre ces étapes pour modifier un utilisateur si l'information a changé.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Lorsque vous créez un utilisateur, celui-ci possède les autorisations par défaut. Si ce compte utilisateur a besoin de permissions supplémentaires, vous devez les lui accorder manuellement après sa création initiale. Si un compte utilisateur a été créé via la fonction d'auto-enregistrement, seul un sous-ensemble des informations sur l'utilisateur est disponible. les informations complémentaires doivent être ajoutées par un utilisateur avec droits d'administration sur le dispositif. Procédez comme suit pour modifier un utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez au panneau **Utilisateurs**.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, accédez à **Collectivité** > **Utilisateurs**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Créer des utilisateurs** dans la section **Etape 1 : configurez l'appareil**.
2. Cliquez sur le ***nom d'utilisateur*** de l'utilisateur que vous désirez modifier. Le nom d'affichage et le nom d'utilisateur ne peuvent pas être modifiés une fois que l'utilisateur a été créé.
3. Vous pouvez éditer le mot de passe et l'adresse électronique de l'utilisateur. Pour modifier le mot de passe, cliquez sur le bouton **[éditer]** de la zone. Entrez un nouveau mot de passe pour modifier le mot de passe.
4. Visualisez l'activité de l'utilisateur. Vous pouvez visualiser les activités suivantes de l'utilisateur à partir de l'écran du compte utilisateur :
    - **Statut actuel** : Cette zone affiche le statut de l'utilisateur. La liste suivante répertorie les statuts possibles :
        - : utilisateur actif au cours des cinq dernières minutes
        - : utilisateur inactif depuis plus de cinq minutes
        - : utilisateur non connecté
    - **Groupes d'utilisateurs** : Cette zone répertorie tous les groupes d'utilisateurs dont l'utilisateur fait partie.

      Entrez un nom de groupe pour ajouter un utilisateur à ce groupe. Quand vous entrez un nom de groupe d'utilisateurs, une liste de groupes d'utilisateurs proches de votre saisie apparaît. Vous devez cliquer sur le nom du groupe d'utilisateurs approprié pour ajouter l'utilisateur à ce groupe. Il ne suffit pas de taper le nom du groupe d'utilisateurs pour y ajouter l'utilisateur. Quand vous ajoutez un utilisateur à un groupe d'utilisateurs, l'utilisateur reçoit les droits d'accès affectés à ce groupe d'utilisateurs. Le niveau de droits d'accès précédent de l'utilisateur n'est pas conservé. Si vous désirez supprimer un utilisateur dans un groupe d'utilisateurs, cliquez sur le lien **[supprimer]** situé en regard du groupe d'utilisateurs dont vous voulez exclure l'utilisateur. Si vous supprimez un utilisateur de tous les groupes d'utilisateurs (en plus du groupe Tout le monde), l'utilisateur conserve les droits d'accès affectés au dernier groupe dont il a été exclu.
5. Modifiez les autorisations de cet utilisateur. Vous pouvez sélectionner ou désélectionner ces autorisations pour contrôler le niveau d'accès affecté à un utilisateur. Vous ne pouvez pas modifier les autorisations d'un utilisateur s'il fait partie d'un groupe, hormis le groupe Tout le monde. Si un utilisateur est membre d'un groupe, il possède les droits d'accès affectés à ce groupe. Si un utilisateur est membre de plusieurs groupes, il possède tous les droits d'accès affectés à tous ces groupes. Quand vous modifiez les droits d'accès affectés à un groupe, la modification est propagée à tous les membres de ce groupe. Les autorisations suivantes sont disponibles pour chaque utilisateur :
    - Administration de dispositif
    - Contrôle du dispositif
    - Création d'une grille de données

    Pour plus d'informations sur ces paramètres d'autorisations, reportez-vous à la rubrique Droits utilisateur.

Résultats

Vous avez modifié un compte utilisateur.

Que faire ensuite

Après avoir modifié l'utilisateur, vous pouvez l'ajouter à un groupe d'utilisateurs. Reportez-vous aux rubriques Création d'un groupe d'utilisateurs et Gestion des groupes d'utilisateurs pour plus d'informations sur la création d'un groupe d'utilisateurs et sur l'ajout d'utilisateurs à des groupes. Vous pouvez ajouter une couche de sécurité à votre dispositif en utilisant un serveur LDAP (Lightweight Directory Access Protocol) pour les authentifications. Pour plus d'informations sur la sécurisation de votre dispositif à l'aide d'un serveur LDAP, reportez-vous à la rubrique Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Suppression d'un utilisateur

Un nom d'utilisateur et un mot de passe sont requis pour se connecter à interface utilisateur. Lorsque vous n'avez plus besoin d'un utilisateur, vous pouvez le supprimer de WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Lorsque vous supprimez un utilisateur, toutes les ressources dont il était propriétaire vous sont automatiquement transférées. Procédez comme suit pour supprimer du dispositif un compte utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez au panneau **Utilisateurs**.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance , accédez à **Collectivité** > **Utilisateurs**.
    - Dans la page d'**accueil**, cliquez sur le lien **Créer des utilisateurs** dans la section **Etape 1 : configurez l'appareil**.
2. Cliquez sur le ***<nom_d'utilisateur>*** de celui que vous désirez supprimer.

    Remarque : Il est impossible de supprimer le compte de l'administrateur (xcadmin).

3. Cliquez sur l'icône de suppression ( ) pour lancer la suppression. Un message s'affiche pour vous inviter à confirmer la suppression définitive de cet utilisateur.
4. Cliquez sur **OK**.

Résultats

Vous avez supprimé un compte utilisateur du dispositif.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Enregistrement d'un nouveau compte utilisateur

Vous pouvez créer votre propre compte utilisateur lorsque l'administrateur active l'option **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. Utilisez cette tâche pour créer un compte utilisateur à partir de l'écran de connexion.

Avant de commencer

Pour enregistrer un nouveau compte utilisateur, sélectionnez *Activé* dans la zone **Permettre aux nouveaux utilisateurs de créer leurs propres comptes**. Pour plus d'informations sur l'activation de l'auto-enregistrement des utilisateurs, reportez-vous à la rubrique Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez à l'écran de connexion d'WebSphere DataPower XC10 Appliance.
2. Cliquez sur le bouton **S'enregistrer...** pour lancer la procédure de création d'un compte utilisateur.
3. Entrez un ID de connexion dans la zone **Nom d'utilisateur**. La valeur saisie dans cette zone est utilisée à la fois comme nom d'utilisateur et comme nom d'affichage. Cette zone peut contenir une valeur comportant jusqu'à 64 caractères. Tous les caractères alphanumériques sont admis. Les caractères spéciaux suivants peuvent également être utilisés : *!@#%^*&-+=*

   Important : Si LDAP (Lightweight Directory Access Protocol) est utilisé pour authentifier les utilisateurs, l'utilisateur en cours d'enregistrement doit préalablement exister dans le référentiel LDAP. Pour plus d'informations sur la création d'une configuration LDAP, reportez-vous à la rubrique Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP .

4. Entrez le mot de passe de l'utilisateur dans la zone **Mot de passe**. Vous pouvez utilisez pour le mot de passe les mêmes caractères que pour la zone Nom d'utilisateur. La zone **Mot de passe** doit être définie si un serveur SMTP (Simple Mail Transfer Protocol) ou LDAP est défini. Si SMTP est activé, vous pouvez entrer un mot de passe ou laissez la zone vide et faire envoyer un mot de passe généré à votre adresse électronique. Si LDAP est utilisé pour authentifier les utilisateurs le mot de passe LDAP existant est utilisé et vous n'avez pas à entrer de mot de passe.
5. Rentrez le même mot de passe de l'utilisateur dans la zone **Vérification du mot de passe**. La valeur saisie dans cette zone doit être identique à celle spécifiée dans la zone **Mot de passe**. Si le contenu de ces zones ne correspond pas, un message d'erreur est affiché lorsque vous cliquez sur **Enregistrement** et vous devez corriger cette erreur avant de pouvoir créer l'utilisateur.
6. Entrez une adresse électronique dans la zone **Adresse électronique**. L'adresse électronique est nécessaire lorsqu'un serveur SMTP (Simple Mail Transfer Protocol) est utilisé. Voir Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance pour plus d'informations.
7. Cliquez sur le bouton **S'enregistrer** pour terminer la procédure d'enregistrement.

Résultats

A l'issue de ces étapes, vous aurez enregistré un compte utilisateur que vous pourrez utiliser pour vous connecter à l'interface utilisateur. Par défaut, seules les autorisations de contrôle du dispositif sont affectées.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Création d'un groupe d'utilisateurs

Vous pouvez créer des groupes d'utilisateurs afin de mieux gérer l'accès de vos utilisateurs à des ressources WebSphere DataPower XC10 Appliance spécifiques.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Créez des groupes d'utilisateurs pour affecter rapidement à une collection d'utilisateurs un accès à une ressource ou un groupe de ressources. Les groupes d'utilisateurs sont vides à leur création initiale. Vous devez ajouter manuellement des utilisateurs à chaque nouveau groupe d'utilisateurs. Procédez comme suit pour créer un utilisateur à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez à **Collectivité** > **Groupes d'utilisateurs**.
2. Cliquez sur l'icône Ajouter ( ) pour créer un groupe.
3. Entrez un nom dans la zone Nom de groupe. La valeur de cette zone peut comporter jusqu'à 64 caractères et ne doit pas être vide. Tous les caractères alphanumériques peuvent être utilisés, ainsi que les caractères spéciaux suivants : *!@#%^*&-+=* . Lorsque l'authentification LDAP est configurée, vous ne pouvez ajouter que des groupes qui existent dans LDAP pour la collectivité.
4. Entrez éventuellement des informations supplémentaires dans la zone Description. Cette zone peut être utilisée pour inclure des informations supplémentaires sur le groupe d'utilisateurs.
5. Cliquez sur **OK**.

Résultats

A l'issue de ces étapes, vous disposerez d'un nouveau groupe d'utilisateurs pour vous faciliter la gestion des autorisations de vos utilisateurs WebSphere DataPower XC10 Appliance.

Que faire ensuite

Vous pouvez ajouter des utilisateurs au groupe d'utilisateurs que vous avez créé. Reportez-vous à la rubrique Gestion des groupes d'utilisateurs pour les instructions détaillées sur l'ajout d'utilisateurs à un groupe.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs

Gestion des groupes d'utilisateurs

Lorsque vous créez un groupe d'utilisateurs, ce dernier ne dispose pas de membres. Vous devez ajouter manuellement les utilisateurs au groupe si l'authentification LDAP n'est pas activée.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Suivez la procédure ci-dessous pour ajouter ou supprimer un utilisateur d'un groupe d'utilisateurs à l'aide de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez à **Collectivité** > **Groupes d'utilisateurs**.
2. Cliquez sur un **<nom de groupe>** pour sélectionner un groupe à modifier.
3. Si vous voulez modifier la description du groupe d'utilisateurs, cliquez sur la description existante et entrez les modifications à effectuer.
4. Si vous voulez ajouter un utilisateur au groupe, tapez l'utilisateur à ajouter, puis cliquez sur le **<nom d'utilisateur>**

   A mesure que vous entrez le nom de l'utilisateur, une liste des utilisateurs correspondant à votre saisie s'affiche. Vous devez cliquez sur le nom de l'utilisateur approprié pour l'ajouter au groupe. La seule saisie du nom de l'utilisateur ne l'ajoute pas pour autant au groupe. Quand vous ajoutez un utilisateur à un groupe d'utilisateurs, l'utilisateur reçoit les droits d'accès affectés à ce groupe d'utilisateurs. Le niveau de droits d'accès précédent de l'utilisateur n'est pas conservé.

   Remarque : Si vous avez activé l'authentification LDAP, vous ne pouvez pas modifier la composition d'un groupe dans WebSphere DataPower XC10 Appliance.

5. Modifiez les droits d'accès affectés au groupe.

   Vous pouvez affecter les droits d'accès suivants à un groupe d'utilisateurs :
   - Administration de dispositif
   - Contrôle du dispositif
   - Création de cache de données

   Pour plus d'informations sur ces paramètres d'autorisations, reportez-vous à la rubrique Droits utilisateur.

6. Si vous désirez supprimer un utilisateur d'un groupe, cliquez sur le lien **[supprimer]** en regard de l'utilisateur concerné. Aucune confirmation n'étant nécessaire pour supprimer l'utilisateur, gérez le groupe d'utilisateurs avec précaution. Si vous supprimez un utilisateur de tous les groupes, autre que le groupe Tout le monde, l'utilisateur conserve les autorisations affectées au dernier groupe dont il est supprimé.

Résultats

Vous avez terminé la modification du groupe d'utilisateurs.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Suppression d'un groupe d'utilisateurs

Suppression d'un groupe d'utilisateurs

Vous pouvez supprimer un groupe d'utilisateurs de WebSphere DataPower XC10 Appliance si le groupe d'utilisateurs est devenu inutile.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Procédez comme suit pour supprimer un groupe d'utilisateurs du dispositif en utilisant l'interface utilisateur WebSphere DataPower XC10 Appliance .

Procédure

1. Cliquez sur **Collectivité** > **Groupes d'utilisateurs**.
2. Cliquez sur le ***<nom_groupe_utilisateurs>*** pour sélectionner celui que vous comptez supprimer.

   Remarque : Le groupe Tout le monde ne peut pas être supprimé.

3. Cliquez sur l'icône de suppression ( ) pour supprimer le groupe.
4. Cliquez sur **OK** pour confirmer la suppression du groupe d'utilisateurs sélectionnés.

Résultats

Chaque membre du groupe d'utilisateurs est supprimé du groupe et le groupe lui-même est supprimé.

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
Droits utilisateur

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs

Droits utilisateur

Les droits utilisateur permettent de déterminer quels panneaux sont visibles par chaque utilisateur et les autorisations d'accès dont il dispose pour un objet spécifique.

Les droits attribués aux utilisateurs définissent les tâches d'administration WebSphere DataPower XC10 Appliance qu'ils sont habilités à effectuer. Ces droits déterminent les pages d'administration qui sont affichées. En outre, le contenu de la page **Bienvenue** est généré dynamiquement afin d'afficher aux utilisateurs des contenus différents en fonction de leur niveau d'accès. Lorsque les utilisateurs s'enregistrent pour la première fois, ils sont autorisés à contrôler le dispositif. Un administrateur doit affecter des droits de création de grille de données ou d'administration de dispositif.

Tableau 1. Ecrans visibles pour chaque niveau de droits

	Droits de contrôle du dispositif	**Droits de création de grille de données**	**Droits d'administration du dispositif**
Page Bienvenue	Oui	Oui	Oui
Création de grilles de données	Non	Oui	Oui
Suppression de grilles de données	Non	Oui	Oui
Affichage du menu de contrôle	Oui	Non	Oui
Affichage des tâches	Oui	Oui	Oui
2.5+ Suppression des tâches terminées	Non	Non	Oui
Affichage et création des collectivités et des zones	Non	Non	Oui
Modification des paramètres du dispositif	Non	Non	Oui
Modification des utilisateurs et des groupes	Non	Non	Oui
2.5+ Visualisation de la gestion des données	Oui	Non	Oui
2.5+ Visualisation de Message Center	Oui	Non	Oui

Rubrique parent : Gestion des utilisateurs et des groupes

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Mot de passe xcadmin

Tâches associées:
Création d'un utilisateur
Gestion des utilisateurs
Suppression d'un utilisateur
Enregistrement d'un nouveau compte utilisateur
Création d'un groupe d'utilisateurs
Gestion des groupes d'utilisateurs
Suppression d'un groupe d'utilisateurs
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Référence associée:
Passerelle REST : Configuration de la sécurité

Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

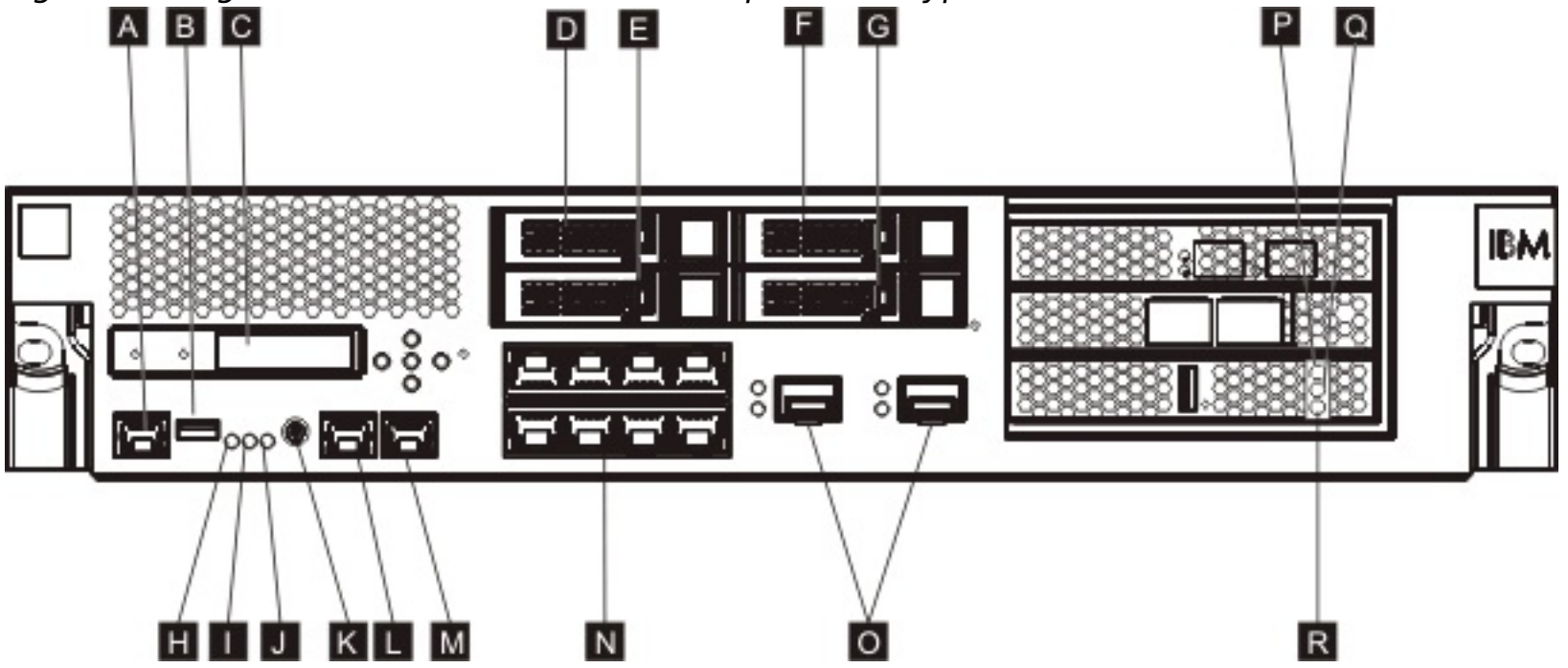
Lors de la connexion série initiale, vous avez configuré l'interface Ethernet *mgmt* afin de connecter le dispositif à votre réseau. Vous pouvez définir des ports Ethernet privés supplémentaires dans l'interface utilisateur.

Avant de commencer

Vous devez disposer des droits d'accès d'administrateur du dispositif.

Pourquoi et quand exécuter cette tâche

Figure 1. Diagramme de la face avant du dispositif de type 7199-92x



- mgmt0 (L)
- mgmt1(M)
- Gauche - Modules Ethernet 1 gigabit eth0-eth7 (N)
- Droite - Modules Ethernet 10 gigabits eth8-eth9 (O)

Pour le dispositif de type 7199-92x, utilisez les ports Ethernet 1 gigabit ou 10 gigabits pour votre grille de données. Vous ne pouvez pas utiliser à la fois des ports 1 gigabit et des ports 10 gigabits. Vous ne pouvez pas changer de type de port après la configuration initiale. Connectez le port de gestion à `MGMT0` (L).

Si vous modifiez les interfaces Ethernet pour un dispositif autonome, vous devez effacer la configuration et redémarrer le dispositif après avoir modifié les paramètres. Si le dispositif fait partie d'une collectivité, vous ne pouvez pas mettre à jour les interfaces Ethernet.

Procédure

1. Modifiez les interfaces Ethernet sur un dispositif autonome. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.
2. Développez l'entrée **Interfaces Ethernet**.
3. Activez ou désactivez une interface Ethernet en cochant ou en désélectionnant la case **Activée**. L'interface *mgmt* ne peut pas être désactivée.
4. Modifiez le **masque/l'adresse IP**. Entrez l'adresse IP et le masque de sous-réseau sous le format suivant *<adresse_ip>/<masque_sous-réseau>*. Le masque de sous-réseau doit être indiqué en respectant la notation CIDR (Classless Inter-Domain Routing). Exemple : *255.255.255.0* en notation longue devient *24* en notation CIDR.
5. Modifiez la **passerelle par défaut**. Le dispositif utilise un routage basé sur la source et non sur la destination. Un paquet est envoyé vers sa destination via l'interface par laquelle il est arrivé. Chaque interface dispose de sa propre table de routage, qui est distincte de celle des autres interfaces. Pour chaque interface qui doit atteindre des destinations au-delà du sous-réseau local, indiquez un chemin par défaut accessible directement depuis cette interface.
6. Indiquez si l'adresse IP fournie est une **Adresse IP privée**.
7. Modifiez l'**unité de transmission maximale (MTU)**. Cette zone spécifie la taille maximale, en octets, d'une unité de données du protocole lors d'une communication via une interface Ethernet. Sa valeur par défaut est de 1500 octets, ce qui est également la valeur maximale autorisée pour cette zone.

8. Modifiez le **mode**. Les modes suivants sont disponibles pour vos interfaces Ethernet :
    - AUTO (Automatique)
    - 10baseT-HD
    - 10baseT-FD
    - 100baseTx-HD
    - 100baseTx-FD
    - 1000baseTx-FD
9. Effacez la configuration, puis redémarrez le dispositif en cochant la case **Activée**. Cette opération efface la configuration. Le redémarrage est nécessaire pour que les processus WebSphere DataPower XC10 Appliance se lient à l'interface Ethernet.

Que faire ensuite

Pour surveiller l'état de vos interfaces Ethernet, voir Surveillance du statut des interfaces Ethernet à l'aide d'une connexion série.

Ajout d'une interface d'agrégation
Vous pouvez configurer une interface d'agrégation dans WebSphere DataPower XC10 Appliance.

Modification d'une interface d'agrégation
Vous pouvez ajouter ou supprimer des membres d'une interface d'agrégation et modifier des propriétés telles que l'adresse IP, la passerelle par défaut ou l'unité de transmission maximale (MTU). Vous pouvez également modifier la règle d'agrégation.

Suppression d'une interface d'agrégation
Vous pouvez supprimer une interface d'agrégation.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Ajout d'une interface d'agrégation

Vous pouvez configurer une interface d'agrégation dans WebSphere DataPower XC10 Appliance.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de WebSphere DataPower XC10 Appliance pour pouvoir réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Vous pouvez lier plusieurs interfaces réseau à une interface d'agrégation unique. Une interface d'agrégation est constituée d'une ou plusieurs interfaces Ethernet servant d'unité logique unique. Cette fonction permet de répartir le trafic sur votre réseau. Vous pouvez utiliser l'interface de ligne de commande ou la console Web pour créer une interface d'agrégation. Il est recommandé d'utiliser l'interface de ligne de commande. Avant chaque opération, vous devez vous connecter au dispositif. Après l'opération, exécutez la commande **`clear-all`** et redémarrez le dispositif.

L'agrégation de liaison entre les commutateurs augmente la connectivité et la bande passante et renforce la redondance.

- Bande passante accrue

  Plusieurs interfaces Ethernet sont regroupées en une seule liaison logique, sous forme d'interface d'agrégation, ce qui augmente la bande passante disponible pour le dispositif.

  A faire : Avec le dispositif, vous pouvez activer des ports de 1 Gbps ou de 10 Gbps. Par conséquent, si vous activez des ports Ethernet de 10 Gbps, vous pouvez ensuite agréger également des ports de 10 Gbps.

- Basculement automatique

  Le trafic sur un port Ethernet défaillant d'une agrégation est transféré vers un port Ethernet actif. L'utilisation d'un commutateur unique offre une protection contre les incidents de port. L'utilisation de commutateurs redondants offre une protection contre les incidents de commutateur réseau.

- Equilibrage de charge

  Vous pouvez sélectionner des règles d'équilibrage de charge pour répartir le trafic entrant et sortant.

- Prise en charge de la redondance

  Deux systèmes peuvent être connectés par plusieurs liaisons dans des configurations d'agrégation différentes.

- Administration améliorée

  En tant qu'administrateur, vous pouvez gérer plusieurs interfaces Ethernet sous la forme d'une unité d'interface d'agrégation unique.

Procédure

- Ajout d'une interface d'agrégation à l'aide de l'interface de ligne de commande. Aidez-vous de l'exemple suivant pour ajouter l'interface d'agrégation :

```
Console> create aggregate-interface myAgg
Console aggregate-interface:myAgg> member eth3 eth4
Console aggregate-interface:myAgg> ip
CWZBR02205I: Entering "ip" mode
Console aggregate-interface:myAgg ip> address 10.5.5.5/24
Console aggregate-interface:myAgg ip> exit
Console aggregate-interface:myAgg> primary-member eth3
Console aggregate-interface:myAgg> aggregation-policy balance-tlb
Console aggregate-interface:myAgg> show
aggregate-interface myAgg:
name "myAgg"
AdminState "Enabled"
use-arp "true"
mtu "1500"
ip
  use-dhcp "false"
  address "10.5.5.5/24"
```

```
  use-slaac "false"
  dad-transmits "1"
  dad-retransmit-timer "1000"
end ip
aggregation-policy "balance-tlb"
lacp-selection-logic "stable"
transmit-hash-policy "layer2"
member "eth3" "eth4"
primary-member "eth3"

Console aggregate-interface:myAgg> exit
```

- Ajout d'une interface d'agrégation à l'aide de la console Web. Affichez le panneau Paramètres de l'une des manières suivantes :
  - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
  - Dans le panneau d'accueil, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.

1. Développez l'entrée **Interfaces d'agrégation**.
2. Cliquez sur **Ajouter une nouvelle interface d'agrégation**.
3. Complétez le formulaire pour décrire l'interface d'agrégation que vous souhaitez ajouter.

   Nom

   Indique le nom de l'interface.

   Adresse/masque IP

   Indique l'adresse IP que vous souhaitez affecter à l'interface d'agrégation. Entrez l'adresse IP et le masque de sous-réseau sous le format suivant *<adresse_ip>/<masque_sous-réseau>*.

   Conseil : Les adresses IPv4 et IPv6 sont toutes deux affectées à l'aide de cette propriété. Les adresses en double ne sont pas admises.

   Règle d'agrégation

   active-backup
   Indique la règle active-backup pour la haute disponibilité. A l'aide de cette règle, une seule interface Ethernet, en tant que membre d'une interface d'agrégation, est activée à la fois. Si cette interface Ethernet échoue, un autre membre reprend le traitement.
   LACP
   Indique la règle LACP (Link Aggregation Control Protocol) pour la haute disponibilité et la bande passante. Vous ne pouvez utiliser la règle LACP que lorsque son mode n'est pas paramétré sur 'OFF'. Par défaut, la logique de sélection est `Stable` et le mode de hachage de transmission est `layer2`.
   balance-tlb
   Indique la règle balance-tlb pour l'équilibrage de charge et la haute disponibilité. Cette règle distribue le trafic sortant en fonction de la charge en cours de chaque membre. Le trafic entrant est transmis à l'interface Ethernet sélectionnée comme membre principal. Si l'interface Ethernet de réception échoue, un autre membre reprend le traitement.

   Membres

   Indique l'interface Ethernet que vous souhaitez ajouter à l'interface d'agrégation. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre principal.

   Restriction : Une interface Ethernet ne peut pas être membre de plusieurs interfaces d'agrégation.

   Important : Assurez-vous que l'interface Ethernet est désactivée avant de l'ajouter à l'interface d'agrégation. Pour désactiver l'interface Ethernet, vous devez désélectionner la case à cocher **Activée** sur le panneau des interfaces Ethernet. Pour savoir comment accéder à ce panneau, voir Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

4. Cliquez sur **OK**.

   Avertissement : Une interface d'agrégation est activée par défaut. Pour continuer d'utiliser l'interface d'agrégation existante à la place de la nouvelle, vous devez d'abord désactiver cette dernière. Pour désactiver l'interface d'agrégation, développez Interfaces d'agrégation, puis recherchez la nouvelle interface d'agrégation que vous venez d'ajouter. Désélectionnez la case **Activée**. Si l'interface d'agrégation reste activée, lors du prochain redémarrage de WebSphere

DataPower XC10 Appliance, la nouvelle interface d'agrégation servira à associer les grilles de données, ce qui risque d'entraîner des résultats imprévus.

Remarque : Vous devez redémarrer pour que les modifications que vous apportez aux interfaces d'agrégation puissent être utilisées par WebSphere DataPower XC10 Appliance. Cependant, vous devez exécuter la commande `clear-all` puis la commande `device-restart` pour que ces modifications prennent effet.

Rubrique parent : Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

Tâches associées:
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Modification d'une interface d'agrégation

Vous pouvez ajouter ou supprimer des membres d'une interface d'agrégation et modifier des propriétés telles que l'adresse IP, la passerelle par défaut ou l'unité de transmission maximale (MTU). Vous pouvez également modifier la règle d'agrégation.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'interface de ligne de commande ou la console Web pour modifier une interface d'agrégation. Il est recommandé d'utiliser l'interface de ligne de commande. Avant chaque opération, vous devez vous connecter au dispositif. Après l'opération, exécutez la commande **`clear-all`** et redémarrez le dispositif.

Procédure

- Modification d'une interface d'agrégation à l'aide de l'interface de ligne de commande. Aidez-vous de l'exemple suivant pour modifier l'interface d'agrégation :

```
Console aggregate-interface:myAgg> aggregation-policy lacp
Console aggregate-interface:myAgg> transmit-hash-policy layer3+4
Console aggregate-interface:myAgg> exit
Console> show aggregate-interface myAgg
aggregate-interface myAgg: [Up]

name "myAgg"
AdminState "Enabled"
use-arp "true"
mtu "1500"
ip
  use-dhcp "false"
  address "10.5.5.5/24"
  use-slaac "false"
  dad-transmits "1"
  dad-retransmit-timer "1000"
end ip
aggregation-policy "lacp"
lacp-selection-logic "stable"
transmit-hash-policy "layer3+4"
member "eth3" "eth4"
primary-member "eth3"
```

- Modification d'une interface d'agrégation à l'aide de la console Web. Affichez le panneau Paramètres de l'une des manières suivantes :
  - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
  - Dans le panneau d'accueil, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.

  1. Développez l'entrée **Interfaces d'agrégation**.
  2. Vous pouvez ajouter ou supprimer des membres dans une interface d'agrégation existante. Une liste d'interfaces Ethernet membres de cette agrégation s'affiche.

     Conseil : Vous ne pouvez pas supprimer de membre s'il s'agit du seul membre, et vous ne pouvez pas supprimer un membre s'il est le membre principal.

     Important : Assurez-vous que l'interface Ethernet est désactivée avant de l'ajouter à l'interface d'agrégation. Pour plus d'informations sur l'activation ou la désactivation d'une interface Ethernet, voir Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance.

     - Pour ajouter d'autres membres, cliquez sur **Ajouter d'autres...**
     - Sélectionnez l'interface Ethernet, puis cliquez sur **OK**.
  3. Modifiez l'adresse IP et le masque de sous-réseau au format suivant : *<adresse_ip>/<masque_sous-réseau>*. Le masque de sous-réseau doit être indiqué en respectant la notation CIDR (Classless Inter-Domain Routing). Par exemple : *255.255.255.0* en notation longue devient *24* en notation CIDR.
  4. Modifiez la **passerelle par défaut**. Le dispositif utilise un routage basé sur la source et non sur la destination. Un paquet est envoyé vers sa destination via l'interface par laquelle il est arrivé.

Chaque interface dispose de sa propre table de routage, qui est distincte des autres interfaces. Pour chaque interface devant atteindre des destinations au-delà du sous-réseau local, indiquez une route par défaut accessible directement à partir de cette interface.

5. Modifiez l'**unité de transmission maximale (MTU)**. Cette zone spécifie la taille maximale, en octets, d'une unité de données du protocole lors d'une communication via une interface Ethernet. Sa valeur par défaut est de 1500 octets, ce qui est également la valeur maximale autorisée pour cette zone.
6. Cliquez sur **Editer** pour modifier la règle d'agrégation. En fonction de la règle d'agrégation à modifier, les propriétés ci-dessous s'affichent.

Règle d'agrégation

active-backup
Indique la règle active-backup pour la haute disponibilité. A l'aide de cette règle, une seule interface Ethernet (en tant que membre d'une interface d'agrégation) est activée à la fois. Si cette interface Ethernet échoue, un autre membre prend le relais. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre principal.
Membre principal
Indique l'interface Ethernet que vous souhaitez désigner comme membre principal. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre principal.

Règle d'agrégation

LACP
Indique la règle LACP (Link Aggregation Control Protocol) pour la haute disponibilité et la bande passante. Vous ne pouvez utiliser la règle LACP que lorsque son mode n'est pas paramétré sur 'OFF'. Par défaut, la logique de sélection est `Stable` et le mode de hachage de transmission est `layer2`.

Logique de sélection

- `Stable`

  Indique l'interface Ethernet ayant la bande passante la plus élevée. Lorsque la valeur stable est sélectionnée, l'interface Ethernet est de nouveau sélectionnée lorsqu'une interface d'agrégation activée ne possède aucun membre disponible. Stable est la valeur par défaut.

- `Bande passante`

  Indique l'interface Ethernet ayant la bande passante la plus élevée. Cette interface Ethernet est de nouveau sélectionnée lorsqu'un autre membre est ajouté ou supprimé ou que l'interface d'agrégation est activée ou désactivée.

- `Nombre`

  Indique l'interface d'agrégation ayant le nombre le plus élevé d'interfaces Ethernet comme membres.

Règle de hachage

- `layer2`

  Indique le résultat OU exclusif (XOR) des adresses MAC afin de générer un hachage.

- `layer2+3`

  Indique le résultat XOR des adresses MAC et des adresses IP afin de générer un hachage.

- `layer3+4`

  Indique le résultat XOR des adresses IP et des numéros de port afin de générer un hachage.

Règle d'agrégation

balance-tlb
Indique la règle balance-tlb pour l'équilibrage de charge et la haute disponibilité. Cette règle distribue le trafic sortant en fonction de la charge en cours de chaque membre. Le trafic entrant est systématiquement transmis à l'interface Ethernet sélectionnée comme membre principal. Si l'interface Ethernet de réception échoue, un autre membre prend le relais.
Membre principal
Indique l'interface Ethernet que vous souhaitez désigner comme membre principal. Par défaut, la première interface Ethernet ajoutée dans la liste est désignée comme membre

principal.

7. Cliquez sur **OK**.

   Important : Lorsque vous modifiez une interface d'agrégation, les modifications ne prennent effet que lorsque vous redémarrez le dispositif. Ce redémarrage est nécessaire pour que les processus WebSphere DataPower XC10 Appliance se lient à l'interface Ethernet. Vous devez désactiver l'interface d'agrégation modifiée avant de redémarrer WebSphere DataPower XC10 Appliance. Pour désactiver l'interface, développez **Interfaces d'agrégation**, puis recherchez l'interface d'agrégation que vous venez de modifier. Désélectionnez la case **Activée**. Après avoir désactivé l'interface d'agrégation, vous devez exécuter les commandes `clear-all` et `device-restart` dans l'interface de ligne de commande. Si vous n'exécutez pas ces commandes, les modifications ne prennent pas effet.

Rubrique parent : Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

Tâches associées:
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Suppression d'une interface d'agrégation

Vous pouvez supprimer une interface d'agrégation.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'interface de ligne de commande ou la console Web pour supprimer une interface d'agrégation. Il est recommandé d'utiliser l'interface de ligne de commande. Avant chaque opération, vous devez vous connecter au dispositif. Après l'opération, exécutez la commande **`clear-all`** et redémarrez le dispositif.

Procédure

- Suppression d'une interface d'agrégation à l'aide de l'interface de ligne de commande. Aidez-vous de l'exemple suivant pour supprimer l'interface d'agrégation :

```
Console> delete aggregate-interface myAgg
Console> list aggregate-interface
```

- Suppression d'une interface d'agrégation à l'aide de la console Web. Affichez le panneau Paramètres de l'une des manières suivantes :
  - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
  - Dans le panneau d'accueil, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : Configurer le dispositif**.

  1. Cliquez sur l'icône de suppression ( ) pour supprimer l'interface d'agrégation.

     Important : Lorsque vous supprimez une interface d'agrégation, les modifications ne prennent effet que lorsque vous redémarrez le dispositif. Après avoir supprimé l'interface d'agrégation, vous devez exécuter les commandes `clear-all` et `device-restart` dans l'interface de ligne de commande. Si vous n'exécutez pas ces commandes, WebSphere DataPower XC10 Appliance continue à utiliser cette interface.

Rubrique parent : Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance

Tâches associées:
Modification d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Importation et exportation de configurations

Lorsque vous configurez un nouveau dispositif, vous pouvez exporter et stocker les paramètres de configuration pour ce dispositif. Si ultérieurement des modifications sont effectuées qui nécessitent la suppression ou la réinstallation du dispositif, vous pouvez alors importer les données de configuration stockées sans perdre les paramètres de configuration.

Avant de commencer

Pour importer ou exporter des configurations, vous devez tout d'abord configurer l'environnement de votre dispositif avec des grilles de données, des utilisateurs ou des groupes, ainsi que des informations LDAP. Pour plus d'informations, voir Configuration de votre dispositif.

Pourquoi et quand exécuter cette tâche

En tant qu'administrateur, vous pouvez configurer des dispositifs, ce qui implique de créer des utilisateurs et des groupes pour la grille de données ainsi que des paramètres de configuration LDAP. Une fois cette configuration effectuée, vous pouvez conserver ces informations en exportant ces configurations dans un fichier. Ensuite, vous pouvez utiliser ce fichier exporté ultérieurement pour réimporter dans le dispositif les informations de configuration exportées. Vous pouvez importer et exporter les informations de configuration suivantes :

- Configurations de grille de données
- Configurations utilisateur
- Configuration de groupe
- Configurations LDAP

Vous devrez peut-être importer et exporter les données de configuration si des erreurs se produisent qui nécessitent que vous supprimiez un dispositif d'une collectivité ou que vous réinstalliez un dispositif dans un environnement autonome. Par exemple, lorsque vous exportez des informations de configuration depuis un dispositif que vous ajoutez à une collectivité, vous pouvez utiliser la fonction d'importation pour récupérer la configuration du dispositif avant que ce dernier rejoigne la collectivité. Dans les environnements autonomes ou de collectivité, l'utilisation des actions d'importation et d'exportation vous permettent de gagner du temps puisque vous n'avez pas à recréer manuellement les informations de configuration pour le dispositif supprimé.

Pour les environnements autonomes : Si vous utilisez un dispositif autonome, effectuez les étapes 1 et 4, qui incluent la syntaxe de commande **`config`** suivante pour l'importation et l'exportation de configurations :

```
config <import|export> -file <nom_fichier> [-silent]
```

La commande **`config export`** accepte également les paramètres dont la valeur est 0. Dans ce cas, la commande effectue une exportation archivée, que vous ne pouvez pas spécifier à l'aide de l'indicateur `-file`.

Pour plus d'informations sur l'utilisation des commandes, entrez `config usage`.

Procédure

1. Exporter vos informations de configuration dans un fichier que vous pouvez utiliser ultérieurement. Par exemple, effectuez une exportation dans le fichier `foo.json`. Utilisez l'interface de ligne de commande, puis exécutez la commande **`config`**, suivie d'une liste des paramètres possibles, par exemple :

   ```
   config <export> -file foo.json -silent
   ```

   Dans l'exemple précédent, le paramètre **`-silent`** est facultatif. Si vous précisez l'indicateur **`-silent`**, les messages d'état de configuration ne s'affichent pas à l'écran (ces messages sont affichés par défaut).
2. Facultatif : Ajouter le nouveau dispositif à une collectivité.
3. Supprimer le dispositif de la collectivité lorsqu'une erreur se produit qui endommage vos données de configuration. Vous pouvez utiliser les outils d'analyse de journal pour analyser le fonctionnement de l'environnement d'exécution et résoudre les problèmes qui se produisent dans l'environnement de votre dispositif.
4. Importer le fichier `foo.json` créé lors de la première étape. Utilisez l'interface de ligne de commande, puis exécutez la commande **`config`**, suivie d'une liste des paramètres possibles, par exemple :

   ```
   config <import> -file foo.json -silent
   ```

Dans l'exemple précédent, le paramètre **-silent** est facultatif. Si vous précisez l'indicateur **-silent**, les messages d'état de configuration ne s'affichent pas à l'écran, ces messages étant affichés par défaut.

5. Créer une nouvelle collectivité avec ce dispositif comme principal.
6. Ajouter d'autres dispositifs à la nouvelle collectivité, un par un.

2.5+ **Exportation de configurations**

Vous pouvez spécifier quand les configurations doivent être exportées à partir de WebSphere DataPower XC10 Appliance.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Exportation de configurations

Vous pouvez spécifier quand les configurations doivent être exportées à partir de WebSphere DataPower XC10 Appliance.

Pourquoi et quand exécuter cette tâche

Les configurations peuvent être exportées selon une périodicité mensuelle, hebdomadaire ou quotidienne, et vous pouvez même préciser l'heure à laquelle elles doivent avoir lieu. Par exemple, vous pouvez spécifier qu'une configuration de grille de données doit être exportée le premier jour du mois, et que les configurations de ce type doivent être conservées dans l'historique pendant une durée d'un an.

Le nombre maximum de configurations exportées pouvant être stockées est de 10. Ce total inclut les exportations créées manuellement et celles créées à l'aide de la console Web.

Procédure

1. Dans la console Web, cliquez sur **Dispositif** > **Importation et exportation de configuration**.
2. Développez la section Export.
    a. Sélectionnez **Planifier les exportations**.
    b. Spécifiez la date et l'heure de début de l'exportation.
    c. Spécifiez la durée de validité de ce planning.
3. Cliquez sur le **bouton de mise à jour du planning**.
4. Facultatif : Si vous voulez exporter une configuration entre deux exportations planifiées, vous devez l'exporter manuellement. Pour ce faire, cliquez sur le **bouton d'exportation immédiate**.

Rubrique parent : Importation et exportation de configurations

Gestion du serveur DNS (Domain Name System)

Un serveur DNS (Domain Name System) est requis pour le système IBM® WebSphere DataPower XC10 Appliance étant donné que les services de recherche DNS sont utilisés pour la communication. Vous devez spécifier ce serveur DNS lors de l'initialisation du dispositif.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Un serveur DNS est requis pour que le dispositif fonctionne correctement. Vos serveurs DNS doivent comporter des entrées DNS directes et inverses pour la plage d'adresses DNS gérées par WebSphere DataPower XC10 Appliance. WebSphere DataPower XC10 Appliance utilise le nom d'hôte dérivé de la recherche inversée lors du déploiement d'un système virtuel. Si cette recherche échoue du fait qu'aucun nom d'hôte n'a été défini, le déploiement ne peut pas aboutir, car il nécessite un nom d'hôte et pas seulement une adresse

Procédure

1. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Serveurs de noms de domaine**.
3. Facultatif : Cliquez sur son **<adresse IP>** pour modifier un serveur DNS existant.
4. Sélectionnez **Cliquez pour ajouter** afin d'ajouter un nouveau serveur DNS. Un serveur DNS est configuré lors de l'initialisation du dispositif.
5. Facultatif : Cliquez sur l'icône de suppression ( ) pour supprimer un serveur DNS.

Résultats

A l'aboutissement de ces étapes, vous aurez défini un serveur DNS à utiliser pour les recherches lors des communications.

Que faire ensuite

Vous pouvez également définir des serveurs DNS avec l'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.

Rubrique parent : Configuration de votre dispositif

Concepts associés:
Mot de passe xcadmin

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Mappage d'adresses IP vers des noms d'hôte

Avant que les informations relatives à une adresse puissent être utilisées pour créer une connexion dans un protocole de réseau TCP/IP, l'adresse IP doit être associée à un nom d'hôte. Vous pouvez résoudre une adresse IP en nom d'hôte en éditant le fichier etc/hosts du dispositif. Vous pouvez éditer le fichier etc/hosts à partir de l'interface utilisateur.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Editez le fichier etc/hosts à partir de l'interface utilisateur de WebSphere DataPower XC10 Appliance.

Procédure

1. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez **Adresses IP vers noms d'hôte**.
3. Cliquez sur **Créer un mappage** pour spécifier l'adresse IP et le nom d'hôte.
4. Cliquez sur **OK** pour créer le mappage. Ce mappage édite le fichier etc/hosts du dispositif.
5. Facultatif : Pour supprimer un mappage, cliquez sur l'icône de suppression ( ).

Résultats

A l'issue de ces étapes, vous aurez édité le fichier etc/hosts qui associe une adresse IP à un nom d'hôte.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Gestion des paramètres de date et heure

Utilisez des serveurs NTP (Network Time Protocol) pour maintenir la synchronisation de la date et de l'heure entre les différents dispositifs de votre collectivité.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Il est important de configurer un serveur NTP lorsque vous utilisez une collectivité. Ce serveur permet d'établir une corrélation entre les journaux des différents dispositifs de la collectivité et donc de garantir la cohérence des dates et des heures au niveau des entrées de journal.

Procédure

1. Accédez au panneau Paramètres. Pour gérer vos paramètres de date et d'heure, accédez au panneau Paramètres à l'aide d'une des méthodes suivantes :
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Date et heure**.
3. Sélectionnez la valeur appropriée dans le menu déroulant **Le fuseau horaire actuel est**.
4. Sélectionnez sur **Cliquez pour ajouter** afin d'ajouter un nouveau serveur NTP. Par défaut, aucun serveur NTP n'est configuré.
5. Cliquez sur le nom du serveur et faites-le glisser pour réorganiser vos serveurs NTP. Le premier serveur NTP disponible dans la liste sera utilisé pour maintenir la synchronisation.
6. Cliquez sur l'icône de suppression ( ) pour supprimer un serveur NTP.
7. Redémarrez le dispositif pour que vos modifications prennent effet. Pour plus d'informations, voir Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Résultats

Vous avez défini un serveur NTP pour maintenir les horloges synchronisées dans les dispositifs de la collectivité.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance

La fonction de distribution du courrier du dispositif est utilisée afin de redéfinir les mots de passe des utilisateurs. Lorsqu'un utiliseur demande un nouveau mot de passe, il le reçoit dans un courrier électronique généré par le dispositif.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Vous devez configurer un serveur SMTP (Simple Mail Transfer Protocol) pour son utilisation avec WebSphere DataPower XC10 Appliance. La fonction de distribution du courrier est utilisée pour envoyer un nouveau mot de passe à l'utilisateur en cas d'oubli. Si l'utilisateur a oublié son mot de passe alors qu'un serveur SMTP n'a pas été configuré, l'utilisateur ne peut pas recevoir un nouveau mot de passe.

Procédure

1. Accédez au panneau Paramètres.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Distribution du courrier**.
3. Ajoutez un serveur SMTP. Indiquez l'adresse IP ou le nom d'hôte du serveur SMTP à utiliser pour WebSphere DataPower XC10 Appliance. Si un nom d'hôte est utilisé pour cette zone, il doit pouvoir être résolu par les serveurs DNS (Domain Name System) définis pour le dispositif. Pour plus d'informations sur l'ajout d'un serveur DNS, reportez-vous à la rubrique Gestion du serveur DNS (Domain Name System).
4. Ajoutez une adresse de réponse. Utilisez pour cette zone l'adresse électronique de l'administrateur.

Résultats

Vous avez spécifié un serveur SMTP et une adresse de réponse à utiliser pour la réinitialisation des mots de passe. Un lien **Mot de passe oublié ?** s'affiche dans l'écran de connexion pour que les utilisateurs puissent réinitialiser leurs mots de passe.

Rubrique parent : Configuration de votre dispositif

Concepts associés:
Mot de passe xcadmin

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur

Le système WebSphere DataPower XC10 Appliance peut être redémarré ou arrêté à partir de l'interface utilisateur.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur du dispositif pour réaliser ces opérations. Lorsque vous arrêtez ou que vous redémarrez le dispositif, vous devez dans un premier temps vérifier que tous les processus en cours d'exécution sont terminés. Pour contrôler la progression des processus en cours d'exécution, cliquez sur **Tâches**.

Pourquoi et quand exécuter cette tâche

Vous pouvez arrêter ou redémarrer WebSphere DataPower XC10 Appliance à partir de l'interface utilisateur.

Procédure

1. Accédez au panneau **Paramètres**. Pour gérer la mise sous tension de votre dispositif, accédez au panneau **Paramètres** à l'aide d'une des méthodes suivantes :
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Paramètres**.
    - Dans la barre de menus située dans la partie supérieure de l'interface utilisateur WebSphere DataPower XC10 Appliance, sélectionnez **Dispositif** > **Identification et résolution des incidents**.
    - Dans le panneau d'**accueil**, cliquez sur le lien **Personnaliser les paramètres** dans la section **Etape 1 : configurez l'appareil**.
2. Développez l'entrée **Alimentation**.
3. Cliquez sur **Redémarrer le dispositif** pour le redémarrer. Pendant le redémarrage, tous les logiciels du dispositif sont arrêtés, puis le dispositif est redémarré. Vous pouvez choisir de redémarrer le dispositif immédiatement ou attendre que toutes les tâches actives soient terminées avant de redémarrer le dispositif.
4. Cliquez sur **Arrêter le dispositif** pour arrêter celui-ci. Pendant l'arrêt, tous les logiciels du dispositif sont arrêtés, puis le dispositif est arrêté. Vous pouvez choisir d'arrêter le dispositif immédiatement ou attendre que toutes les tâches actives soient terminées avant de l'arrêter. Pour interrompre l'alimentation du dispositif, vous devez actionner l'interrupteur d'alimentation physique placé au dos du dispositif afin d'éteindre celui-ci.

Résultats

2.5+ Vous pouvez suivre l'état de démarrage du dispositif dans l'interface utilisateur. Pour plus d'informations, voir Suivi de l'état de démarrage du dispositif.

A l'issue de cette procédure, l'appareil sera arrêté ou redémarré, en fonction de votre sélection.

Rubrique parent : Configuration de votre dispositif

Tâches associées:
Sécurité
Gestion des utilisateurs et des groupes
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Importation et exportation de configurations
Gestion du serveur DNS (Domain Name System)
Mappage d'adresses IP vers des noms d'hôte
Gestion des paramètres de date et heure
Gestion de la distribution du courrier de IBM WebSphere DataPower XC10 Appliance
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Contrôle des activités à l'aide des tâches

Configuration des collectivités et des zones

Les collectivités et les zones permettent de définir la topologie physique (emplacements) de vos dispositifs WebSphere DataPower XC10 Appliance.

1. Création d'une collectivité
   Pour regrouper plusieurs WebSphere DataPower XC10 Appliance à des fins de disponibilité, d'évolutivité et de gestion, vous devez créer une collectivité.

2. Création et modification des zones
   Les zones permettent d'indiquer l'emplacement physique d'un dispositif. Cet emplacement physique peut correspondre à deux emplacements différents dans le même centre de données ou à des dispositifs situés dans des centre de données différents, éventuellement dans différentes régions du pays.

3. Configuration d'une réplication multimaître entre les collectivités
   La réplication multimaître est une technique qui permet de garantir une disponibilité continue pour plusieurs environnements de déploiement. En établissant des liaisons de collectivité entre les serveurs de catalogue dans vos collectivités, vous pouvez répliquer des données de manière asynchrone entre les grilles de données de deux collectivités différentes.

4. 2.5+ Configuration d'IBM eXtremeIO (XIO)
   IBM® eXtremeIO (XIO) est un mécanisme de transport qui remplace ORB (Object Request Broker).

Création d'une collectivité

Pour regrouper plusieurs WebSphere DataPower XC10 Appliance à des fins de disponibilité, d'évolutivité et de gestion, vous devez créer une collectivité.

Avant de commencer

Avant de définir une collectivité, vous devez posséder les droits d'accès d'administration du dispositif sur les consoles des dispositifs que vous souhaitez ajouter à la collectivité.

Pourquoi et quand exécuter cette tâche

Pour créer une collectivité, vous devez ajouter tous les noms d'hôte et clés secrète de tous les dispositifs à regrouper dans une collectivité à la configuration de l'un des dispositifs. Les dispositifs peuvent communiquer toutes les informations de configuration. Ces informations incluent les grilles de données, les informations de contrôle, les utilisateurs et les groupes et les appartenances à la collectivité.

Tous vos dispositifs doivent être au même niveau de microprogramme avant que vous puissiez les ajouter à une collectivité. Pour plus d'informations sur l'exécution des mises à niveau de microprogramme, voir Mise à jour du microprogramme.

Procédure

1. Sélectionnez un dispositif auquel vous allez ajouter les noms d'hôtes et les clés secrètes des autres dispositifs de la collectivité. Pour tous les autres dispositifs, ouvrez une nouvelle fenêtre de navigateur et effectuez la procédure suivante :
   a. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres**.
   b. La liste des dispositifs de la collectivité s'affiche. Le dispositif auquel vous êtes connecté est signalé avec l'icône de console ( ). Notez le **nom d'hôte** et **la clé secrète** de ce dispositif.

   Tableau 1. Etats de dispositif

Icône	Définition
	Indique le dispositif auquel vous êtes connecté et que le dispositif est actif.
	Indique que le dispositif est démarré et qu'il peut être configuré et utilisé.
	Indique que le dispositif est arrêté et qu'il n'est pas disponible.

2. Ajoutez tous les noms d'hôtes et les clés secrètes de chaque dispositif à l'un des dispositifs du groupe.
   a. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres**.
   b. Cliquez sur l'icône Ajouter ( ).
   c. Accédez à **Nom d'hôte** et **Clé secrète** pour le dispositif à ajouter à la collectivité.
3. Contrôlez la progression de l'ajout des dispositifs à la collectivité dans le vue Tâches. Dans l'interface utilisateur, cliquez sur **Tâches**.

Résultats

Les dispositifs sont regroupés dans une collectivité afin de faciliter l'évolutivité et la gestion.

Rubrique parent : Configuration des collectivités et des zones
Rubrique suivante : Création et modification des zones

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Création et modification des zones

Les zones permettent d'indiquer l'emplacement physique d'un dispositif. Cet emplacement physique peut correspondre à deux emplacements différents dans le même centre de données ou à des dispositifs situés dans des centre de données différents, éventuellement dans différentes régions du pays.

Avant de commencer

Avant de créer et de configurer des zones, vous devez installer, configurer et assembler vos dispositifs au sein d'une collectivité. Vous pouvez alors sélectionner les dispositifs de votre collectivité et les répartir dans des zones spécifiques. Si vous ajoutez un dispositif à la collectivité après la création des zones, vous pouvez spécifier la zone lors de l'ajout du dispositif.

Pourquoi et quand exécuter cette tâche

En spécifiant des zones de manière à indiquer l'emplacement physique des dispositifs, les données stockées dans les grilles de données peuvent être placées à l'emplacement approprié. DataPower XC10 Appliance utilise des zones pour déterminer l'emplacement des informations principales et secondaires du cache. Par exemple, si une grille de données principale se trouve dans la zone 1, la réplique de cette grille de données se trouve en zone 2. Si vous n'avez qu'une seule zone, aucune réplique n'est créée.

Lors de la première initialisation d'un dispositif, le dispositif est ajouté à la zone par défaut appelée **DefaultZone**. Lorsque vous ajoutez les dispositifs aux zones que vous créez, ils sont supprimés de la zone **DefaultZone**. Vous ne pouvez pas remplacer les dispositifs dans la zone **DefaultZone** après les en avoir retirés.

Procédure

1. Dans l'interface utilisateur, cliquez sur**Collectivité** > **Zones**.
2. Ajoutez des zones. Cliquez sur **Ajouter des zones**. Indiquez le nom de la zone à ajouter. Le nom de la zone s'affiche sur la page de la console. Vous pouvez répéter cette procédure si vous devez créer plusieurs zones.
3. Ajoutez des dispositifs à la zone. Développez le nom de la zone et sélectionnez les dispositifs à ajouter à la zone. Répétez cette procédure jusqu'à ce que tous les dispositifs de la collectivité aient été ajoutés aux zones. Chaque dispositif doit se trouver dans une zone. Une même zone ne peut pas contenir de dispositifs de collectivités différentes.
4. Appliquez les modifications de zone. Les modifications apportées aux zones sont effectives lorsque vous cliquez sur **Activer les modifications de zone**. Pour modifier les informations de zone, vous pouvez cliquer sur l'icône de suppression ( ) pour supprimer une zone ou cliquer sur **Réinitialiser les modifications de zone** pour restaurer la dernière configuration de zone activée.

Résultats

DataPower XC10 Appliance peut créer des zones de cache principales et secondaires en fonctions des emplacements physiques (zones) définis. Vous pouvez afficher ou modifier la topologie des zones à tout moment en cliquant sur **Collectivité** > **Zones** dans l'interface utilisateur.

Que faire ensuite

Créez et configurez vos applications et vos grilles de données.

Rubrique parent : Configuration des collectivités et des zones
Rubrique précédente : Création d'une collectivité
Rubrique suivante : Configuration d'une réplication multimaître entre les collectivités

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Configuration d'une réplication multimaître entre les collectivités

La réplication multimaître est une technique qui permet de garantir une disponibilité continue pour plusieurs environnements de déploiement. En établissant des liaisons de collectivité entre les serveurs de catalogue dans vos collectivités, vous pouvez répliquer des données de manière asynchrone entre les grilles de données de deux collectivités différentes.

Avant de commencer

- Au moins deux collectivités doivent être configurées. Pour que les données soient répliquées entre les collectivités, vous devez créer des configurations de grille de données identiques dans chaque collectivité.
- Déterminez le type de topologie de collectivité multiple que vous souhaitez définir. Pour plus d'informations sur les topologies multimaîtres, voir Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître.
- Vous pouvez configurer des liaisons entre les collectivités dans l'interface utilisateur ou à l'aide de l'utilitaire **xscmd**.

Pourquoi et quand exécuter cette tâche

Une seule collectivité ne couvre pas un réseau non fiable car des incidents de faux positif risquent d'être détectés. Cependant, vous pouvez être amené à répliquer des données de grille de données sur des dispositifs dont la connectivité réseau n'est pas fiable. Voici quelques scénarios courants dans lesquels vous pouvez être amené à utiliser ce type de topologie :

- Reprise après incident entre des centres de données dans lesquels une collectivité est active et une autre est utilisée aux fins de secours.
- Centres de données géographiquement répartis dans lesquels toutes les collectivités sont actives pour les clients géographiquement proches.

Une fois que vous connectez deux collectivités, toutes les grilles de données portant le même nom sont répliquées de façon asynchrone d'une collectivité à l'autre. Ces grilles de données doivent comporter le même nombre de répliques dans chaque collectivité et posséder les mêmes configurations de mappe dynamique.

Procédure

1. Etablissez une liaison entre les collectivités.
   - **Etablissement de liaisons de collectivité à l'aide de l'interface utilisateur :**
     a. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Liaisons de collectivité**.
     b. Cliquez sur l'icône d'ajout () et entrez le nom d'hôte ou l'adresse IP, le nom d'utilisateur et le mot de passe du dispositif distant.
   - **Etablissement de liaisons de collectivité à l'aide de l'utilitaire `xscmd` :**
     a. Extrayez l'adresse IP et le port des serveurs de catalogue dans chaque collectivité. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres** > ***nom_membre_collectivité***. Une liste de serveurs de catalogue et de numéros de port s'affiche. Répétez cette étape pour chaque collectivité à connecter.
     b. Connectez l'utilitaire **xscmd** à l'une des collectivités. Pour plus d'informations sur le démarrage de l'utilitaire **xscmd**, voir Administration avec l'utilitaire xscmd.

        ```
        xscmd.sh -ts xsatruststore.jks -tst jks -ssl -tsp xc10pass
        -user xcadmin -tsp xcadmin -cep myxc10.mycompany.com:2809
        [additional_xscmd_parameters]
        ```

     c. A partir de la collectivité à laquelle vous êtes connecté, exécutez la commande suivante :

        ```
        xscmd –c establishLink -cep myxc10.mycompany.com:2809 -fd dname
        -fe myxc102.mycompany.com:2809,myxc103.mycompany.com:2809
        ```

        Le paramètre **-fd dname** spécifie le nom de la collectivité éloignée avec laquelle vous voulez établir une liaison. Dans l'exemple ci-dessus, le nom de la collectivité éloignée est **dname**. Pour trouver le nom d'une collectivité, consultez la liste des serveurs de catalogue sur le panneau des informations détaillées sur les membres. Le nom de la collectivité est la première adresse IP de la liste.

Une fois la liaison établie, les serveurs de catalogue des collectivités commencent à se répliquer mutuellement. Il n'est pas nécessaire d'établir la liaison dans les deux sens.

2. Vous pouvez également annuler des liaisons entre les collectivités.
    - **Annulation de liaisons de collectivité à l'aide de l'interface utilisateur :** Dans l'interface utilisateur, cliquez sur l'icône de suppression () pour supprimer une liaison sélectionnée.
    - **Annulation de liaisons de collectivité à l'aide de l'utilitaire `xscmd` :** A l'aide de l'utilitaire **`xscmd`**, exécutez la commande suivante :

      ```
      xscmd –c dismissLink -cep myxc10.mycompany.com:2809 -fd dname
      ```

Rubrique parent : Configuration des collectivités et des zones
Rubrique précédente : Création et modification des zones
Rubrique suivante : 2.5+ Configuration d'IBM eXtremeIO (XIO)

Concepts associés:
Topologies pour assocation de collectivités pour la mise en oeuvre d'une réplication multimaître
Topologie du dispositif : collectivités, zones et grilles de données

Affichage du type de transport du domaine de service de catalogue

Vous pouvez afficher le type de transport qui est actuellement utilisé pour le domaine de service de catalogue.

Avant de commencer

Vous pouvez afficher les types de transports qui sont utilisés dans un domaine de service de catalogue autonome ou un domaine de service de catalogue en cours d'exécution dans WebSphere Application Server.

- Si vous utilisez un domaine de service de catalogue autonome, utilisez l'utilitaire **xscmd** pour afficher les informations de transport sur le domaine de service de catalogue. Pour plus d'informations sur la configuration de l'utilitaire **xscmd**, voir Administration avec l'utilitaire xscmd.
- Si vous disposez d'un domaine de service de catalogue qui s'exécute dans WebSphere Application Server, vous pouvez utiliser l'utilitaire **wsadmin** pour afficher le type de transport. Pour plus d'informations sur l'utilitaire **wsadmin**, voir Démarrage du client de scriptage wsadmin en utilisant le scriptage wsadmin.

Procédure

- Affichez le type de transport d'un domaine de service de catalogue autonome. Dans l'utilitaire **xscmd**, exécutez la commande suivante :
  - UNIX `./xscmd.sh -c showTransport`
  - Windows `xscmd.bat -c showTransport`

  La commande affiche le type de transport. Les valeurs suivantes peuvent s'afficher : `eXtremeIO` ou `Object Request Broker`.
- Affichez le type de transport d'un domaine de service de catalogue qui s'exécute WebSphere Application Server.
  - Dans la console d'administration, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaine de service de catalogue** > ***nom_domaine_service_catalogue***. Vérifiez que **Activer IBM eXtremeIO (XIO) communication** est sélectionné.
  - Dans l'utilitaire **wsadmin**, exécutez la commande suivante :
    - A l'aide de Jacl :

      ```
      $AdminTask getTransport {-domainName TestDomain }
      ```

    - A l'aide de la chaîne Jython :

      ```
      AdminTask.getTransport ('[-domainName testDomain]')
      ```

    La commande affiche le type de transport. Les valeurs suivantes peuvent s'afficher : `eXtremeIO` ou `Object Request Broker`. Si vous exécutez cette commande sur un domaine de service de catalogue qui contient des serveurs distants ou que catalogServerName est un serveur distant, une erreur se produit. Vous devez utiliser la commande **xscmd -c showTransport** pour les serveurs distants. Pour plus d'informations sur la commande **getTransport** dans l'utilitaire **wsadmin**, voir Tâches d'administration des domaines de service de catalogue.

Rubrique parent : 2.5+ Configuration d'IBM eXtremeIO (XIO)

Configuration des grilles de données

Vous pouvez créer trois types différents de grilles de données : des grilles de données simples grilles de données, de session grilles de données et de mémoire cache dynamique.

Création de grilles de données simples
Les grille de données simples permettent d'effectuer des opérations de création, de récupération, de mise à jour et de suppression. Une grille de données simple permet notamment d'accélérer l'accès aux données sur une base de données.

Création de grilles de données de session
Vous pouvez créer des grilles de données pour enregistrer des sessions HTTP à partir de vos applications Java ou .NET.

Java **Création de grilles de données en cache dynamique**
IBM® WebSphere DataPower XC10 Appliance permet de stocker des données pour une instance de cache dynamique WebSphere Application Server. En configurant cette fonction, vous permettez aux applications qui utilisent une instance de cache dynamique WebSphere Application Server d'utiliser les fonctions et les fonctionnalités de performance du dispositif.

Configuration de la capacité maximale d'une grille de données
Vous pouvez définir une capacité maximale pour chaque grille de données de la collectivité. La configuration d'une capacité maximale limite la quantité de stockage de données qui peut être utilisée par une grille de données. La limite de capacité permet de s'assurer que la capacité de stockage disponible pour la collectivité est utilisée de façon prévisible.

Activation de la sécurité pour les grilles de données
Par défaut, la sécurité est désactivée pour chacune des grilles de données que vous créez. Vous pouvez modifier les paramètres de sécurité d'une grille de données de manière à en restreindre l'accès à certains utilisateurs ou groupes d'utilisateurs.

Effacement des grilles de données
Vous pouvez supprimer définitivement toutes les entrées d'une grille de données. Vous pouvez effacer la grille de données pour supprimer les informations périmées ou tester les entrées.

Suppression des grilles de données
Si vous souhaitez supprimer les données d'une grille de données, vous pouvez supprimer la grille de données puis la recréer.

Configuration d'un fournisseur de cache Spring
Spring Framework Version 3.1 a introduit une nouvelle abstraction de cache. Celle-ci vous permet d'ajouter de manière transparente la mise en cache à une application Spring existante. Vous pouvez utiliser WebSphere DataPower XC10 Appliance comme fournisseur de cache pour l'abstraction de cache.

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Création de grilles de données simples

Les grille de données simples permettent d'effectuer des opérations de création, de récupération, de mise à jour et de suppression. Une grille de données simple permet notamment d'accélérer l'accès aux données sur une base de données.

Avant de commencer

Les grilles de données simples peuvent s'utiliser avec WebSphere Application Server ou avec une application Java™ autonome. Le client WebSphere eXtreme Scale Client doit être installé dans tous les cas.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser les grilles de données simples pour accélérer les applications Web dynamiques Web en allégeant la charge de la base de données. Vous pouvez stocker les paires clé-valeur de données arbitraires en mémoire, réduisant ainsi les requêtes de base de données lourdes. Les clés peuvent correspondre à n'importe quel type Java java.lang.String ou Entier. Les valeurs peuvent correspondre à n'importe quel type d'objet. A chaque fois que des données doivent être utilisées, la grille de données simple du dispositif est consultée en premier. Si les données ne se trouvent pas sur le dispositif, elles sont récupérées à partir de la base de données, puis insérées dans la grille de données.

Procédure

1. Créez la grille de données simple. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Grille de données simple**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données simple grille de données à créer. Les caractères suivants ne peuvent pas être utilisés dans le nom de la grille de données : `^ . \\ / , # $ @ : ; \ * ? < > | = + & % [ ] " "`.
2. Téléchargez le fichier `objectgrid.xml` pour votre grille de données simple. Dans la configuration de la grille de données simple que vous avez créée, cliquez sur l'icône de téléchargement ( ) et enregistrez le fichier sur votre disque local.
3. Créer une application qui accède à la grille de données. Pour plus d'informations, voir Développement d'applications pour accéder à des grilles de données simples.

Que faire ensuite

- Configurez la sécurité avant de commencer à envoyer des données à la grille de données. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
- Configurez des répliques. Les répliques permettent de s'assurer que les données de vos grilles de données sont disponibles si la copie principale échoue. Pour configurer des répliques, cliquez sur **Grille de données** > **Grille de données simple** > **Afficher les attributs avancés**. Les répliques ne sont créées que si le dispositif est dans une collectivité. Si le nombre de dispositifs dans la collectivité est *n*, le nombre maximal de répliques est *n-1*. Ainsi, si vous configurez trois répliques mais que vous n'avez que deux dispositifs dans la collectivité, une seule réplique est créée. Des répliques supplémentaires sont créées si vous ajoutez des dispositifs à la collectivité. Définissez le nombre de réplique au nombre idéal de répliques souhaitées : ainsi, lorsque des dispositifs rejoignent la collectivité, de nouvelles répliques sont créées. Le contenu de la grille de données est effacé lorsque vous modifiez le nombre de répliques.
- Configurez une limite de capacité pour la grille de données. En configurant des limites de capacité sur la grille de données, vous vous assurez que la capacité de stockage pour la collectivité est utilisée de façon prévisible. Pour plus d'informations, voir Configuration de la capacité maximale d'une grille de données.
- Configurez un expulseur TTL pour une grille simple. Pour plus d'informations, voir Configuration d'un expulseur TTL (Time To Live).
- Vous pouvez contrôler votre grille de données dans l'interface utilisateur de DataPower XC10 Appliance. Pour plus d'informations, voir Surveillance des grilles de données dans l'interface utilisateur.

Configuration des mappes dynamiques
Vous pouvez créer de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Configuration d'une stratégie de verrouillage
Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale. Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste.

Configuration d'un expulseur TTL (Time To Live)

Lorsque vous créez une grille simple, une mappe par défaut (statique) et un ensemble de mappes dynamiques sont créés. Par défaut, aucun expulseur TTL n'est configuré pour une mappe par défaut. Si vous disposez d'une mappe dynamique, vous pouvez définir une valeur de durée de vie (TTL) pour la date/heure de création (.*CT), la date/heure de la dernière mise à jour (*.LUT) ou la date/heure du dernier accès (*.LAT). Vous pouvez modifier ce comportement par défaut de sorte qu'un expulseur TTL soit également activé pour une mappe par défaut.

Java **Configuration du cache local**

Vous pouvez configurer vos clients pour un cache local en ligne L'on appelle cache local ce cache facultatif. Il s'agit d'une grille de données indépendante, présente sur chaque client et faisant office de cache du cache distant côté serveur. Le cache local est activé par défaut lorsque le verrouillage est désactivé ou configuré comme optimiste et ne peut pas être utilisé lorsqu'il est défini comme pessimiste.

2.5+ **Gestion des noms d'hôte pour la communication entre les clients et les serveurs**

Utilisez la propriété **`publishHost`** pour configurer le nom d’hôte que le transport par réseau publie à destination des serveurs et clients connectés.

Rubrique parent : Configuration des grilles de données

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Configuration des mappes dynamiques

Vous pouvez créer de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Avant de commencer

- Créez une grille de données simple. Pour plus d'informations, voir Création de grilles de données simples.
- Choisissez les options de configuration que vous souhaitez utiliser dans votre mappe dynamique. Pour plus d'informations, voir Options de configuration de mappe dynamique.

Que faire ensuite

Créez une mappe dynamique à partir de vos modèles définis :

- Java **Avec des API Java** Voir Création de mappes dynamiques avec les API Java pour un exemple d'appel de la Session.getMap(String) pour définir votre mappe dynamique.
- .NET 2.5+ **Avec des API .NET** Voir Création de mappes dynamiques avec les API .NET pour plus d'informations.
- **Avec la passerelle REST :** Voir Exemple de passerelle REST : Création de mappes dynamiques pour plus d'informations.

Rubrique parent : Création de grilles de données simples

Tâches associées:
Configuration d'une stratégie de verrouillage
Configuration d'un expulseur TTL (Time To Live)
Java Configuration du cache local
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs
Java 2.5+ Configuration de l'invalidation du cache local

Référence associée:
Options de configuration de mappe dynamique

Configuration d'une stratégie de verrouillage

Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale. Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste.

Avant de commencer

Déterminez la stratégie de verrouillage à utiliser. Pour plus de détails, voir Stratégies de verrouillage.

Pourquoi et quand exécuter cette tâche

Chaque instance de mappe de sauvegarde peut être configurée pour utiliser l'une de ces stratégies de verrouillage :

- Mode de verrouillage optimiste (valeur par défaut)
- Mode de verrouillage pessimiste (requis pour les applications .NET)
- Aucun

Procédure

Pour savoir comment configurer un mode de verrouillage optimiste, un mode de verrouillage pessimiste ou aucun mode de verrouillage, voir Options de configuration de mappe dynamique.

Rubrique parent : Création de grilles de données simples

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
Configuration des mappes dynamiques
Configuration d'un expulseur TTL (Time To Live)
Java Configuration du cache local
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs
Java Configuration et mise en oeuvre du verrouillage dans les applications Java
Configuration d'une stratégie de verrouillage
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Configuration d'un expulseur TTL (Time To Live)

Lorsque vous créez une grille simple, une mappe par défaut (statique) et un ensemble de mappes dynamiques sont créés. Par défaut, aucun expulseur TTL n'est configuré pour une mappe par défaut. Si vous disposez d'une mappe dynamique, vous pouvez définir une valeur de durée de vie (TTL) pour la date/heure de création (.*CT), la date/heure de la dernière mise à jour (*.LUT) ou la date/heure du dernier accès (*.LAT). Vous pouvez modifier ce comportement par défaut de sorte qu'un expulseur TTL soit également activé pour une mappe par défaut.

Avant de commencer

- Créez une grille de données simple pour votre configuration.
- Décidez sur quel expulseur Time vous souhaitez l'utiliser. Pour plus d'informations sur les types d'expulseur spécifiques, voir Expulseurs.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Grille de données simple** > ***nom_grille_de_données*** > **Afficher les attributs avancés**.
2. Dans la zone de liste **Expulseur pour la mappe intitulée** , sélectionnez un type d'expulseur. Si vous décidez de conserver la valeur NONE pour cette option, l'expulseur TTL est désactivé uniquement pour une mappe par défaut. Dans le cas contraire, le fait de sélectionner un type d'expulseur autre que NONE active TTL pour une mappe par défaut.
3. Indiquez une valeur dans la zone **Expulser les données des mappes TTL (Time To Live) après**. La valeur par défaut TTL est de `3600` secondes. La valeur que vous indiquez ici s'applique également à une mappe par défaut si le type d'expulseur est paramétré sur une valeur autre que **NONE**. La valeur `0` permet de désactiver l'expulsion TTL pour la totalité de la grille.
4. Cliquez sur **Appliquer les modifications** pour enregistrer les modifications. Vous êtes averti de la perte de données une fois que la grille redémarre.

Exemple

Scénario par défaut :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **NONE** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `3600`. Il en résulte que les entrées sont expulsées au bout de 1 heure uniquement pour les mappes dynamiques se terminant par *.CT, *.LUT et *.LAT.

Scénario de personnalisation du type d'expulseur par défaut sur une valeur autre que NONE :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **Heure de création** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `3600`. Il en résulte que les entrées sont expulsées au bout de 1 heure pour les mappes dynamiques se terminant par *.CT, *.LUT et *.LAT. La mappe par défaut expulse les entrées 1 heure après l'heure de création.

Personnalisation du type d'expulseur par défaut et scénario de valeur TTL :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **Heure de la dernière mise à jour** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `4800`. Il en résulte que les entrées sont expulsées au bout de `80` minutes pour les mappes dynamiques se terminant par *.CT, *.LUT et *.LAT. La mappe par défaut expulse les entrées `80` minutes après l'heure de la dernière mise à jour de l'entrée.

Scénario de personnalisation de la valeur TTL uniquement :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **NONE** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `0`. Il en résulte que l'expulsion TTL est désactivée pour toutes les mappes (par défaut et dynamiques) de cette grille.

Scénario de personnalisation du type d'expulseur par défaut et de la valeur TTL :

Dans ce scénario, le type d'expulseur sélectionné dans la zone de liste **Expulseur pour la mappe intitulée** est **Heure du dernier accès** et la valeur TTL indiquée dans la zone **Expulser les données des mappes TTL (Time To Live) après** est `0`. Il en résulte que l'expulsion TTL est désactivée pour toutes les mappes (par défaut et dynamiques) de cette grille.

Expulseurs
Les expulseurs retirent des données de la grille de données. Vous pouvez configurer un expulseur pour une mappe dynamique et une mappe par défaut sur une grille simple.

Rubrique parent : Création de grilles de données simples

Concepts associés:
Expulseurs

Tâches associées:
Configuration des mappes dynamiques
Configuration d'une stratégie de verrouillage
Java Configuration du cache local
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs
Java 2.5+ Configuration de l'invalidation du cache local

Référence associée:
Options de configuration de mappe dynamique

Java

Expulseurs

Les expulseurs retirent des données de la grille de données. Vous pouvez configurer un expulseur pour une mappe dynamique et une mappe par défaut sur une grille simple.

Types d'expulseur

L'expulseur supprime les entrées en se basant sur un concept de durée de vie. Vous pouvez sélectionner un expulseur en fonction de l'heure de sa création, de l'heure de son dernier accès ou sa dernière mise à jour. Par défaut, un expulseur est créé à l'aide d'une mappe dynamique. Pour activer un expulseur sur une mappe par défaut pour une grille simple, voir Configuration d'un expulseur TTL (Time To Live).

Aucun

Spécifie que les entrées n'expirent jamais et par conséquent ne sont jamais supprimées de la mappe.

Heure de création

Spécifie que les entrées sont expulsées en fonction de la date et de l'heure auxquelles elles ont été créées.

Si vous utilisez l'attribut l'expulseur Heure de création , celui-ci expulse une entrée lorsqu'il aura atteint la durée de vie spécifiée par sa valeur TTL , qui est définie en millisecondes dans la configuration de l'application. Si vous paramétrez TTL sur la valeur de 10 secondes, l'entrée est automatiquement expulsée dix secondes après son insertion.

Il est important de faire attention lors de la définition de cette valeur pour le type d'expulseur Heure de création . Cet expulseur s'avère utile dans les cas de quantités raisonnablement élevées d'ajouts au cache utilisés pendant un temps donné. Avec cette stratégie, tout ce qui est créé est supprimé à la fin de la durée définie.

Le type d'expulseur Heure de création est utile dans des scénarios où l'on doit actualiser des cours boursiers toutes les 20 minutes, voire moins. Supposons qu'une application Web récupère les cours de la bourse, sans que l'obtention des cours les plus récents soit cruciale. Dans ce cas, les cours sont mis en cache dans une grille de données pendant 20 minutes. Après 20 minutes, les entrées de la mappe expirent et sont expulsées. Toutes les vingt minutes environ, la grille de données actualise les données à l'aide des données de la base de données. Celle-ci est actualisée toutes les 20 minutes avec les cours les plus récents.

Heure du dernier accès

Spécifie que les entrées sont expulsées en fonction de l'heure de leur dernier accès, qu'elles aient été lues ou actualisées.

Heure de la dernière modification

Spécifie que les entrées sont expulsées en fonction de l'heure de leur dernière modification.

Si vous utilisez le type d'expulseur Heure du dernier accès ou Heure de la dernière modification , paramétrez la valeur TTL sur une valeur plus faible que si vous utilisiez le type d'expulseur Heure de création . Les entrées sont réinitialisées lors de chaque accès. En d'autres termes, si la valeur de est égale à 15 et qu'une entrée a existé pendant 14 secondes mais qu'on y accède, elle n'expire pas de nouveau pendant les 15 prochaines secondes. Si vous paramétrez TTL sur une valeur relativement élevée, il est possible que de nombreuses entrées ne soient expulsées. Cependant, si vous le paramétrez sur une valeur telle que 15 secondes, les entrées avec peu d'accès risquent d'être supprimées.

Le type d'expulseur Heure du dernier accès ou Heure de la dernière mise à jour est utile dans des scénarios où l'on doit conserver les données de session d'un client, à l'aide d'une mappe de grille de données. Les données de session doivent être détruites si le client ne les utilise pas pendant un certain laps de temps. Par exemple, après 30 minutes d'inactivité du client. Dans ce cas, l'utilisation du type d'expulseur Heure du dernier accès ou Heure de la dernière mise à jour avec la valeur TTL paramétrée sur 30 minutes convient tout à fait à cette application.

Rubrique parent : Configuration d'un expulseur TTL (Time To Live)

Tâches associées:
Configuration d'un expulseur TTL (Time To Live)

Référence associée:
Options de configuration de mappe dynamique

Java

Configuration du cache local

Vous pouvez configurer vos clients pour un cache local en ligne L'on appelle cache local ce cache facultatif. Il s'agit d'une grille de données indépendante, présente sur chaque client et faisant office de cache du cache distant côté serveur. Le cache local est activé par défaut lorsque le verrouillage est désactivé ou configuré comme optimiste et ne peut pas être utilisé lorsqu'il est défini comme pessimiste.

Avant de commencer

Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.

Pourquoi et quand exécuter cette tâche

Un cache local est rapide, car il offre un accès en mémoire à un sous-ensemble de l'ensemble des données en cache stockées à distance. Par défaut, le cache local côté client n'a pas de taille maximale et des erreurs de mémoire insuffisante peuvent se produire sur le client. Pour contrôler la taille du cache local, configurez un remplacement côté client qui active un expulseur TTL ou LRU sur le client. Pour plus d'informations sur le remplacement des paramètres client, voir Configuration des clients Java avec une configuration XML et Configuration des clients Java à l'aide d'un programme.

Procédure

Vous pouvez activer la création d'un cache local sur le client par la définition du nom de mappe dynamique. Vous devez définir la mappe dynamique pour avoir un verrouillage optimiste ou aucun verrouillage. Exemples de noms de mappe pour lesquelles un cache local est activé :

```
my_grid.NONE
my_grid.NONE.O
```

Pour plus d'informations, voir Configuration des mappes dynamiques. Pour connaître la liste des options de configuration de mappe dynamique, voir Options de configuration de mappe dynamique.

Résultats

Pour vérifier qu'un cache local est activé, exécutez la méthode BackingMap.isNearCacheEnabled() dans le client. Vous pouvez également rechercher le message CWOBJ1128I dans les fichiers journaux sur le client pour déterminer si le cache local est activé.

Java 2.5+ **Configuration de l'invalidation du cache local**

Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée dans la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Ce mécanisme est plus rapide que l'utilisation de l'expulsion TTL (time to live, durée de vie) dans le cache local.

Rubrique parent : Création de grilles de données simples

Tâches associées:
Configuration des mappes dynamiques
Configuration d'une stratégie de verrouillage
Configuration d'un expulseur TTL (Time To Live)
2.5+ Gestion des noms d'hôte pour la communication entre les clients et les serveurs

Java

Configuration de l'invalidation du cache local

2.5+ Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée dans la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Ce mécanisme est plus rapide que l'utilisation de l'expulsion TTL (time to live, durée de vie) dans le cache local.

Avant de commencer

- Vous devez utiliser IBM eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).
- Vous devez utiliser un cache local. Pour déterminer si le cache local est activé, exécutez la méthode BackingMap.isNearCacheEnabled() dans le client. Pour plus d'informations sur la configuration du cache local, voir Configuration du cache local.
- Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.

Pourquoi et quand exécuter cette tâche

L'activation de l'invalidation dans le cache local fournit un ensemble plus précis de données de la grille de données distante, car le cache local est mis à jour lorsque les données distantes sont modifiées.

Procédure

2.5+ Créez une mappe dynamique qui inclut l'option d'invalidation du cache local. Pour plus d'informations, voir Configuration des mappes dynamiques. Examinez les exemples de mappes suivants, pour lesquels l'invalidation du cache local est activée :

```
my_grid.NCI
my_grid.CT.NCI
my_grid.LAT.NCI
my_grid.LUT.NCI
my_grid.NONE.NCI
my_grid.CT.O.NCI
my_grid.LAT.O.NCI
my_grid.LUT.O.NCI
my_grid.NONE.O.NCI
```

Pour connaître la liste des options de configuration de mappe dynamique, voir Options de configuration de mappe dynamique.

Rubrique parent : Java Configuration du cache local

Tâches associées:
Configuration des mappes dynamiques
Configuration d'un expulseur TTL (Time To Live)

Référence associée:
Options de configuration de mappe dynamique

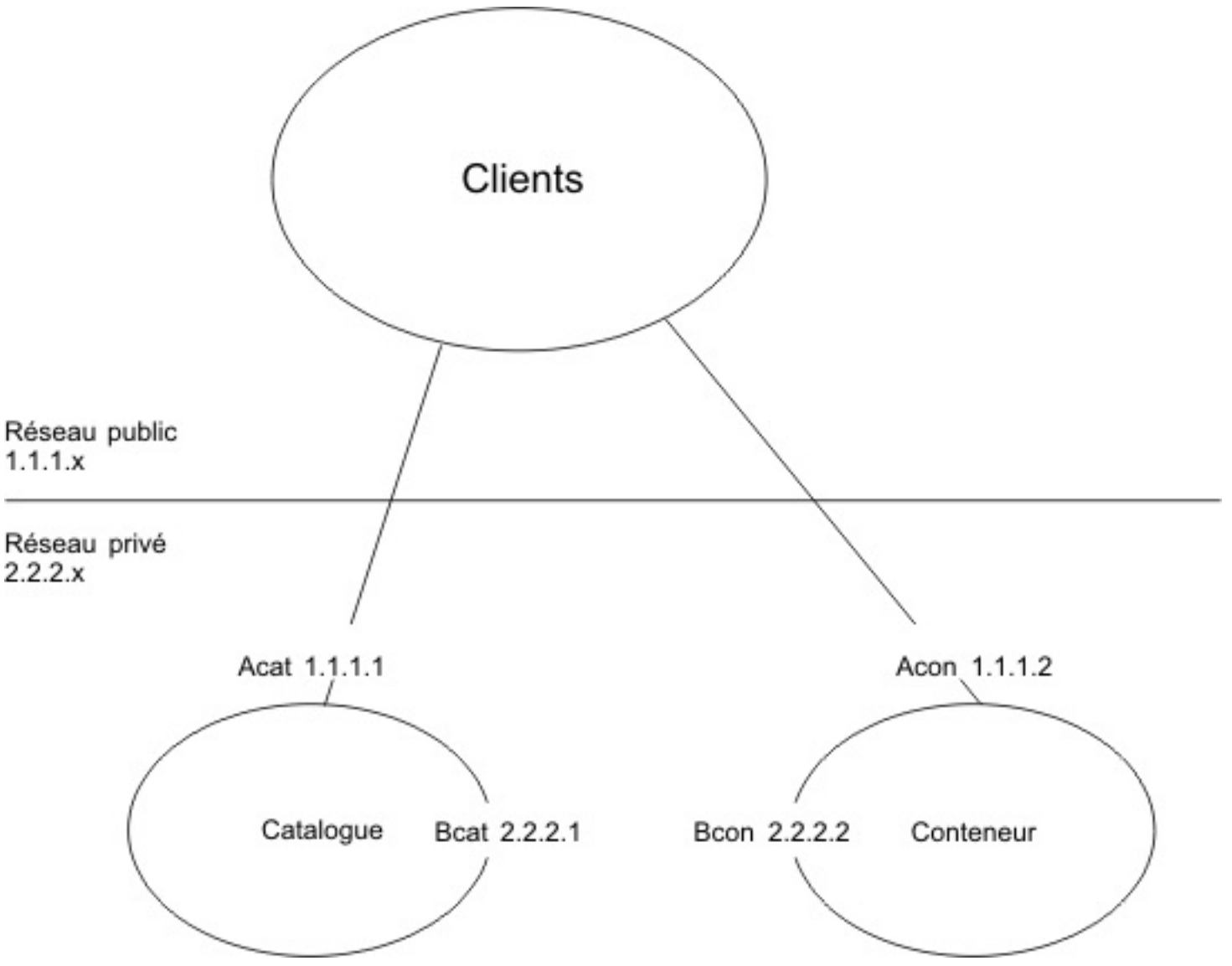
Gestion des noms d'hôte pour la communication entre les clients et les serveurs

Utilisez la propriété **`publishHost`** pour configurer le nom d’hôte que le transport par réseau publie à destination des serveurs et clients connectés.

Pourquoi et quand exécuter cette tâche

Avec le serveur DNS ou le fichier hosts, vous pouvez configurer et redéfinir l'hôte qui est publié à l'aide de la propriété **`publishHost`**. Vous disposez désormais d'un contrôle plus fin sur la communication entre les clients et les serveurs, ce qui vous permet également d'optimiser la communication entre les noeuds dans la grille de données. A titre d'exemple, le diagramme ci-dessous montre comment vous pouvez connecter les clients au protocole IP public (1.1.1.x), et les serveurs au protocole IP privé (2.2.2.x). Dans cet exemple, les serveurs de catalogue et de conteneur ont deux noms d'hôte pour chaque carte réseau : `A*` pour les adresses IP publiquement disponibles et `B*` pour les adresses IP privées.

*Figure 1. Topologie pour la propriété **`publishHost`**.*



Le nom d'hôte `C*` représente la nouvelle valeur de la propriété **`publishHost`**, qui est utilisée comme nom d'hôte générique à publier. Il constitue également une alternative à `A*` ou `B*`, car les noms `A*` ne sont pas appropriés pour les serveurs qui doivent communiquer sur un réseau privé et les noms `B*` ne sont pas appropriés pour les clients qui doivent communiquer sur un réseau public. Il vous permettra de choisir entre les adresses alternatives publique et privée des serveurs.

Procédure

1. Définissez la propriété **`publishHost`** avec la valeur `Ccat` sur le serveur de catalogue et avec la valeur `Ccon` sur le serveur de conteneur. Cette propriété définit le nom d'hôte qui est publié sur le DNS et pendant les communications des clients et des serveurs.
2. Facultatif : Définissez la propriété **`listenerHost`**. Cette propriété contrôle les adresses IP que chaque serveur écoute.
3. Assurez-vous que le DNS (ou les fichiers `/etc/hosts`) des clients qui utilisent le réseau public fait bien référence aux hôtes `Ccat` et `Ccon` à l'aide de leur adresse IP publique (respectivement Ccat 1.1.1.1 et Ccon 1.1.1.2). Assurez-vous que le DNS (ou les fichiers `/etc/hosts`) des serveurs qui utilisent le réseau privé fait bien référence aux hôtes `Ccat` et `Ccon` à l'aide de leur adresse IP privée (respectivement Ccat 2.2.2.1 et Ccon 2.2.2.2).
4. Utilisez la valeur **`publishHost`** du serveur de catalogue (Ccat) comme nom d'hôte des noeuds finaux de serveur de catalogue qui sont utilisés dans la configuration de grille d'objets sur les clients et les serveurs.

Résultats

Désormais, le trafic inter-serveur circule sur le réseau public, et le trafic entre les clients et les conteneurs circule sur le réseau privé.

Rubrique parent : Création de grilles de données simples

Tâches associées:
Configuration des mappes dynamiques
Configuration d'une stratégie de verrouillage
Configuration d'un expulseur TTL (Time To Live)
Java Configuration du cache local

Création de grilles de données de session

Vous pouvez créer des grilles de données pour enregistrer des sessions HTTP à partir de vos applications Java ou .NET.

Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données
Configurez votre application WebSphere Application Server pour qu'elle utilise le dispositif ou la gestion de session. Vous pouvez soit sélectionner le dispositif lorsque vous installez une nouvelle application, soit actualiser votre application existante ou les paramètres du serveur pour qu'ils utilisent le dispositif.

Configuration du gestionnaire de sessions HTTP avec WebSphere Application Server
Alors que WebSphere Application Server offre une fonction de gestion de sessions, les performances se dégradent quand le nombre de demandes augmente. WebSphere eXtreme Scale est fourni avec une mise en oeuvre de la gestion de sessions qui offre la réplication de session, une haute disponibilité, une meilleure évolutivité et des options de configuration plus robustes.

Configuration du gestionnaire de sessions HTTP avec WebSphere Portal
Vous pouvez rendre persistantes des sessions HTTP à partir de WebSphere Portal dans une grille de données.

Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications
WebSphere eXtreme Scale Client est fourni avec une implémentation de gestion de session qui remplace le gestionnaire de sessions par défaut pour un conteneur Web. Cette implémentation offre la réplication de session, la haute disponibilité, une meilleure évolutivité et des options de configuration. Vous pouvez activer le gestionnaire de réplication de session WebSphere eXtreme Scale Client et le démarrage du conteneur intégré ObjectGrid générique pour différents serveurs d'application, tels que Tomcat.

Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Vous pouvez migrer une session de réplication de mémoire à mémoire ou une session de base de données pour utiliser la gestion de session WebSphere eXtreme Scale.

.NET 2.5+ **Configuration d'un fournisseur de stockage d'état de session ASP.NET**
Vous pouvez stocker l'état de session de vos applications ASP.NET dans une grille de données.

2.5+ **Affichage de la taille des sessions HTTP**
Vous pouvez utiliser l'utilitaire **xscmd** pour afficher la taille des sessions dans votre application Java. Cette information peut vous aider à identifier et résoudre les problèmes qui peuvent survenir dans votre grille de données.

Paramètres d'initialisation du contexte de servlet
La liste de paramètres d'initialisation du contexte de servlet suivante peut être spécifiée dans le fichier splicer.properties en fonction de la méthode de raccord choisie.

Fichier splicer.properties
Le fichier `splicer.properties` contient toutes les options de configuration qui permettent de configurer un gestionnaire de sessions basé sur un filtre de servlet.

Rubrique parent : Configuration des grilles de données

Java

Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données

Configurez votre application WebSphere Application Server pour qu'elle utilise le dispositif ou la gestion de session. Vous pouvez soit sélectionner le dispositif lorsque vous installez une nouvelle application, soit actualiser votre application existante ou les paramètres du serveur pour qu'ils utilisent le dispositif.

Avant de commencer

Pour pouvoir modifier la configuration dans WebSphere Application Server, vous devez :

- disposer d'un accès à la cellule WebSphere Application Server à configurer ;
- connaître l'adresse IP ou le nom d'hôte qualifié complet du dispositif ;
- disposer d'un ID utilisateur et d'un mot de passe utilisables pour vous connecter à l'interface utilisateur du dispositif. Pour pouvoir créer une grille de données, vous devez disposer des droits de création d'une mémoire cache de données.
- avoir installé WebSphere eXtreme Scale Client dans la configuration WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client.
- avoir activé la sécurité globale dans la console d'administration WebSphere Application Server. Activez la sécurité globale si la sécurité de la couche de transport est activée sur le dispositif ou si vous voulez que les clients utilisent la sécurité de la couche de transport. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

Procédure

- **Pour configurer la gestion de session lors de l'installation de l'application, effectuez la procédure suivante :**

  1. Dans la console d'administration de WebSphere Application Server, cliquez sur **Applications** > **Nouvelle application** > **Nouvelle application d'entreprise**. Sélectionnez le chemin **Détaillé** pour la création de l'application, puis effectuez les premières étapes de l'assistant.
  2. A l'étape **Paramètres de gestion des sessions eXtreme Scale session** de l'assistant, configurez la grille de données que vous voulez utiliser. Pour la zone **Gérer la persistance des sessions par** , choisissez **WebSphere DataPower XC10 Appliance**. Entrez les informations concernant le dispositif et la grille de données du dispositif que vous voulez utiliser. Vous pouvez créer une grille de données ou utiliser une grille de données existante déjà configurée sur le dispositif.

     Si vous souhaitez sauvegarder vos sessions sur une grille de données existante du dispositif, vous devez connaître le nom de la grille de données à utiliser. Toutefois, vous pouvez créer une grille de données sur le dispositif lors de la configuration de l'application. Pour créer une grille de données de session avant de configurer l'application dans l'interface utilisateur du dispositif, cliquez sur **Grille de données** > **Session**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données de session à créer. Ce nom ne peut contenir les caractères suivants : `^ . \\ / , # $ @ : ; \ * ? < > | = + & % [ ] " "`.
  3. Terminez l'installation de l'application en effectuant la suite de procédure de l'assistant.

Vous pouvez également installer l'application à l'aide d'un script wsadmin. Dans l'exemple suivant, le paramètre**-SessionManagement** crée une configuration identique à celle que vous pouvez créer dans la console d'administration :

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission
.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XC10SessionManagement myXC10.ibm.com:!:username:!:password:!:AGrid80]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host]
[MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-
INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host]
[MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

- **Pour configurer la gestion de session sur une application existante dans la console d'administration de WebSphere Application Server :**

  Remarque : La case **Override session management** est cochée quand l'application est configurée pour utiliser WebSphere DataPower XC10 Appliance. Cela signifie que tout paramètre de session de niveau serveur défini dans la configuration de WebSphere Application Server sera remplacé par les paramètres de session de niveau application. Si ce n'est pas ce que vous souhaitez, vous pouvez activer WebSphere DataPower XC10 Appliance au niveau serveur.

  1. Dans la console d'administration WebSphere Application Server, cliquez sur **Applications** > **Types d'application** > **Applications d'entreprise WebSphere** > ***nom_application*** > **Propriétés du module Web** > **Gestion des sessions** > **Paramètres de gestion des sessions eXtreme Scale**.
  2. Actualisez les champs pour activer la persistance des sessions vers une grille de données.

  Vous pouvez également mettre à jour l'application à l'aide d'un script wsadmin. Dans l'exemple suivant, le paramètre**-SessionManagement** crée une configuration identique à celle que vous pouvez créer dans la console d'administration :

  ```
  AdminApp.edit('A-edition9.0', '[ -SessionManagement [[true
  XC10SessionManagement myXC10.ibm.com:!:username:!:password:!:AGrid80]]]')
  ```

  Les caractères `:!:` envoyés sont utilisés comme délimiteurs. Les valeurs transmises sont les suivantes :

  ```
  ID_application:!:nom_utilisateur:!:mot-de-passe:!:
  nom_grille
  ```

  Lorsque vous enregistrez les modifications, l'application utilise la grille de données configurée pour la persistance des sessions sur le dispositif.

- **Pour configurer la gestion de session sur un serveur existant :**

  1. Dans la console d'administration WebSphere Application Server, cliquez sur **Serveurs** > **types de serveur** > **Serveurs d'applications WebSphere** > ***nom_serveur*** > **Gestion des sessions** > **Paramètres de gestion des sessions eXtreme Scale**.
  2. Actualisez les champs pour activer la persistance des sessions.

  Les commandes wsadmin suivantes vous permettent de configurer également la gestion des sessions sur un serveur existant :

  ```
  AdminTask.configureServerSessionManagement('[-nodeName my_node
  -serverName server1 -enableSessionManagement true -sessionManagementType
  XC10SessionManagement -XC10SessionManagement [-applianceIdentifier myserver.ibm.com
  -userName -password ******** -gridName myTestGrid]]')
  ```

  Lorsque vous enregistrez les modifications, le serveur utilise la grille de données configurée pour la persistance de sessions avec toutes les applications qu'il exécute.

- Si vous souhaitez modifier d'autres aspects de la configuration des sessions HTTP, vous pouvez modifier le fichier `splicer.properties`. Vous pouvez obtenir l'emplacement du chemin du fichier `splicer.properties` en recherchant la propriété personnalisée **sessionFilterProps**. Si vous avez configuré la persistance de session au niveau du serveur, le nom de la propriété personnalisée est `com.ibm.websphere.xs.sessionFilterProps`. Si vous l'avez configurée au niveau de l'application, elle s'appelle *<nom_application>*`,com.ibm.websphere.xs.sessionFilterProps`. Ces propriétés personnalisées peuvent se trouver dans l'un des emplacements suivants :
  - Dans un environnement WebSphere Application Server Network Deployment : Le fichier `splicer.properties` se trouve dans le chemin du profil du gestionnaire de déploiement.
  - Dans un environnement WebSphere Application Server autonome : une propriété personnalisée sur le serveur d'applications

Vous pouvez ouvrir le fichier indiqué, effectuez les modifications et synchroniser les noeuds pour propager le fichier mis à jour des propriétés vers les autres noeuds de la configuration. Tous les noeuds de serveur d'applications nécessitent que le fichier `splicer.properties` se trouve dans le chemin défini pour que les sessions persistent.

Avertissement : Si vous souhaitez activer la persistence pour les sessions qui utilisent la réécriture d'URL, affectez à la propriété **`useURLEncoding`** la valeur `true` dans le fichier `splicer.properties`.

Pour plus d'informations sur les propriétés dans le fichier `splicer.properties`, voir Fichier splicer.properties.

Résultats

Vous avez configuré le gestionnaire de sessions HTTP pour que les sessions soient conservées vers une grille de données. Les entrées sont supprimées de la grille de données lorsque les sessions expirent. Voir Paramètres de gestion des sessions pour plus d'informations sur la mise à jour la valeur de temporisation des sessions dans la console d'administration WebSphere Application Server.

Si l'ensemble de la grille de données qui héberge les données de sessions d'application est inaccessible à partir du client du conteneur Web, le client utilise le conteneur Web de base dans la gestion de sessions WebSphere Application Server. La grille de données peut être inaccessible dans les cas suivants :

- Problème de réseau entre le conteneur Web et le dispositif.
- Arrêt des processus serveur dans le dispositif.

Les sessions les moins utilisées sont invalidés à partir du cache de session du conteneur Web. Si la grille de données dans le dispositif devient disponible, les sessions ayant été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données du dispositif est indisponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données de production lorsque le système est chargé.

ATTENTION :

Lorsque vous configurez ce scénario, les données d'identification de sécurité pour IBM WebSphere DataPower XC10 Appliance sont automatiquement stockées dans la configuration de WebSphere Application Server. Si vous êtes amené à modifier les données d'identification pour la grille de données après cette première configuration, WebSphere Application Server ne disposera plus des données d'identification correctes. Vous pourrez les réinitialiser en appliquant à nouveau les paramètres de gestion de session eXtreme Scale.

Que faire ensuite

- Configurez la sécurité avant de commencer à envoyer des données à la grille de données. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
- Configurez des répliques. Les répliques permettent de s'assurer que les données de vos grilles de données sont disponibles si la copie principale échoue. Pour configurer des répliques, cliquez sur **Grille de données** > **Session** > **Afficher les attributs avancés**. Les répliques ne sont créées que si le dispositif fait partie d'une collectivité. Si le nombre de dispositifs de la collectivité est *n*, le nombre maximal de répliques est *n*-*1*. Ainsi, si vous configurez trois répliques mais que vous n'avez que deux dispositifs dans la collectivité, une seule réplique est créée. Des répliques supplémentaires sont créées si vous ajoutez des dispositifs à la collectivité. Indiquez comme nombre de répliques le nombre idéal de répliques souhaitées : ainsi, lorsque des dispositifs rejoignent la collectivité, de nouvelles répliques sont créées. Le contenu de la grille de données est effacé lorsque vous modifiez le nombre de répliques.
- Configurez une limite de capacité pour la grille de données. En configurant des limites de capacité sur la grille de données, vous vous assurez que la capacité de stockage pour la collectivité est utilisée de façon prévisible. Pour plus d'informations, voir Configuration de la capacité maximale d'une grille de données.
- Vous pouvez surveiller votre grille de données de session dans l'interface utilisateur de DataPower XC10 Appliance. Pour plus d'informations, voir Surveillance des grilles de données dans l'interface utilisateur.

Rubrique parent : Création de grilles de données de session

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Référence associée:
Fichier splicer.properties

Information associée:
Installation de fichiers d'application d'entreprise à l'aide de la console
Installation d'applications d'entreprise à l'aide d'un script wsadmin

Java

Configuration du gestionnaire de sessions HTTP avec WebSphere Application Server

Alors que WebSphere Application Server offre une fonction de gestion de sessions, les performances se dégradent quand le nombre de demandes augmente. WebSphere eXtreme Scale est fourni avec une mise en oeuvre de la gestion de sessions qui offre la réplication de session, une haute disponibilité, une meilleure évolutivité et des options de configuration plus robustes.

Avant de commencer

- Par défaut, le fichier de propriétés du raccordeur (splicer) de XC10 utilise un ID utilisateur et un mot de passe générés par le système. Cet ID est XC10_USER_nomGrille et son mot de passe ne peut pas être modifié. Si vous changez l'ID utilisateur, vous devez manuellement mettre à jour la combinaison ID utilisateur/mot de passe dans le fichier `splicer.properties` en modifiant la propriété `credentialGeneratorProps`. Pour plus d'informations, voir Comment modifier le fichier splicer.properties. Cette combinaison générée par le système n'est utilisée pour obtenir le fichier `splicer.properties` que lors du démarrage, et ces données d'identification ne sont plus utilisées par le client WebSphere Application Server.
- WebSphere eXtreme Scale doit être installé dans votre cellule WebSphere Application Server ou WebSphere Application Server Network Deployment pour que vous puissiez utiliser le gestionnaire de sessions d'eXtreme Scale. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
- Lorsque vous utilisez WebSphere eXtreme Scale pour la réplication de session HTTP sur WebSphere Application Server, l'option **Autoriser la gestion des sessions de dépassement** doit être activée pour chaque application Web concernée et pour le serveur d'applications qui l'héberge. Pour plus d'informations, voir Paramètres de gestion de sessions.
- La sécurité globale doit être activée dans WebSphere Application Server si vous avez activé SSL (Secure Sockets Layer) sur le dispositif. Elle doit également l'être si vous voulez utiliser SSL pour une collectivité qui le prend en charge. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Le gestionnaire de sessions HTTP de WebSphere eXtreme Scale prend en charge les serveurs imbriqués et éloignés pour la mise en cache.

- **Scénario de serveurs imbriqués**

  Dans ce scénario, les serveurs de grille de données sont regroupés dans les processus où les servlets sont exécutés. Le gestionnaire de sessions peut communiquer directement avec l'instance ObjectGrid locale, pour éviter les retards réseau coûteux.

  Si vous utilisez WebSphere Application Server, placez dans les répertoires `META-INF` de vos fichiers d'archive Web (WAR) les fichiers *rép_base_wxs*`/session/samples/objectGrid.xml` et *rép_base_wxs* `/session/samples/objectGridDeployment.xml` fournis. eXtreme Scale détecte automatiquement ces fichiers au démarrage de l'application et démarre automatiquement les conteneurs eXtreme Scale dans le même processus que le gestionnaire de sessions.

  Vous pouvez modifier le fichier `objectGridDeployment.xml`, suivant que vous souhaitiez utiliser une réplication synchrone ou asynchrone et en fonction du nombre de répliques à configurer.

- **Scénario avec serveurs éloignés**

  Dans le scénario avec serveurs éloignés, les serveurs de conteneur et les servlets s'exécutent dans des processus différents. Le gestionnaire de sessions communique avec un serveur de conteneur éloigné. Pour pouvoir utiliser un serveur éloigné connecté au réseau, le gestionnaire de sessions doit être configuré avec les noms d'hôte et les numéros de port du domaine de service de catalogue. Le gestionnaire de sessions utilise ensuite une connexion client eXtreme Scale pour communiquer avec le serveur de catalogues et avec les serveurs de conteneur.

  Si les serveurs de conteneur démarrent dans des processus autonomes indépendants, démarrez les conteneurs de grille de données avec les fichiers `objectGridStandAlone.xml` et `objectGridDeploymentStandAlone.xml` fournis dans le répertoire des exemples du gestionnaire de sessions.

Procédure

1. Raccordez votre application de sorte qu'elle puisse utiliser le gestionnaire de sessions. Pour utiliser le gestionnaire de sessions, vous devez ajouter les déclarations de filtre appropriées aux descripteurs de

déploiement Web de l'application. En outre, les paramètres de configuration du gestionnaire de sessions sont transmis au gestionnaire de sessions sous la forme de paramètres d'initialisation du contexte de servlet dans les descripteurs de déploiement. Ces informations peuvent être injectées dans votre application de différentes manières :

- **Raccord automatique avec WebSphere Application Server**

  Lorsque vous installez votre application, vous pouvez la configurer pour qu'elle utilise le gestionnaire de sessions HTTP pour la grille de données. Vous pouvez également modifier la configuration de l'application ou du serveur pour qu'ils utilisent le gestionnaire WebSphere eXtreme Scale de sessions HTTP. Pour plus d'informations, voir Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données.

- **Raccord automatique de l'application avec des propriétés personnalisées**

  Vous n'avez pas besoin de raccorder manuellement vos applications lorsqu'elles s'exécutent dans WebSphere Application Server ou dans WebSphere Application Server Network Deployment.

  Ajoutez une propriété personnalisée à une cellule ou à un serveur pour définir à cette portée le fichier `splicer.properties` pour toutes les applications Web. Pour configurer la propriété personnalisée, procédez comme suit :

  a. Dans la console d'administration de WebSphere Application Server, accédez au chemin correct de l'emplacement que vous voulez définir dans la propriété personnalisée pour indiquer l'emplacement du fichier `splicer.properties`.
     - Pour définir la propriété personnalisée pour toutes les applications ou pour une application spécifique, cliquez sur **Administration du système** > **Cellule** > **Propriétés personnalisées**.
     - Pour définir la propriété personnalisée à appliquer à toutes les applications sur un serveur d'applications spécifique, cliquez sur **Serveur d'applications** > ***<nom_serveur>*** > **Administration** > **Propriétés personnalisées**. Le nom de la propriété est `com.ibm.websphere.xs.sessionFilterProps` et sa valeur se trouve dans le fichier `splicer.properties` dont a besoin votre application. Exemple de chemin d'emplacement d'un fichier : `/opt/splicer.properties`.
  b. Ajoutez la propriété personnalisée `com.ibm.websphere.xs.sessionFilterProps`. La valeur de cette propriété personnalisée indique l'emplacement du fichier `splicer.properties` à modifier. Le fichier existe dans le gestionnaire de déploiement. Si vous voulez indiquer le fichier `splicer.properties` pour une application spécifique à l'aide d'une propriété personnalisée au niveau de la cellule, entrez le nom de la propriété personnalisée comme `<`*`nom_application`*`>,com.ibm.websphere.xs.sessionFilterProps`, où *`nom_application`* représente le nom de l'application pour laquelle vous voulez appliquer la propriété personnalisée.

  Important : Vérifiez que le fichier `splicer.properties` mis à jour se trouve dans le même chemin sur tous les noeuds contenant un serveur d'applications hébergeant la ou les applications qui sont à raccorder pour la réplication de session.

  Les portées de cellule, de serveur et d'application sont les seules portées disponibles lors d'une exécution dans un gestionnaire de déploiement. Si vous avez besoin d'une autre portée, raccordez manuellement vos applications Web.

  A faire : Notez aussi que le raccordement automatique ne fonctionne que si tous les noeuds exécutant l'application contiennent le fichier `splicer.properties` dans le même chemin. Dans le cas d'environnements mixtes contenant des noeuds Windows et UNIX, cette manière de procéder n'est pas disponible et vous devez raccorder manuellement l'application.

- **Raccorder l'application avec le script `addObjectGridFilter`**

  Utilisez un script de ligne de commande fourni avec eXtreme Scale pour raccorder une application avec des déclarations de filtre et une configuration sous forme de paramètres d'initialisation de contexte de servlet. Pour un déploiement WebSphere Application Server, ce script se trouve dans `<répertoire_principal_was>/optionalLibraries/ObjectGrid/session/bin/addObjectGridFilter.bat/sh`. Pour un déploiement autonome, il se trouve dans `REP_PRINCIPAL_WXS/ObjectGrid/session/bin/addObjectGridFilter.sh/bat`. Le script **`oddObjectGridFilter`** utilise deux paramètres :
  - Application : chemin absolu du fichier archive d'entreprise à raccorder
  - Chemin absolu du fichier de propriétés du raccordeur qui contient des propriétés de configuration.

Voici le format du script :

Windows

```
addObjectGridFilter.bat [fichier_ear]
[fichier_propriétés_raccordeur]
```

UNIX

```
addObjectGridFilter.sh [fichier_ear]
[fichier_propriétés_raccordeur]
```

UNIX **Exemple utilisant eXtreme Scale installé sur WebSphere Application Server sur UNIX :**

a. `cd` *`rép_base_wxs`*`/optionalLibraries/ObjectGrid/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear`
   `racine_was``/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Exemple utilisant eXtreme Scale installé dans un répertoire autonome sur UNIX :**

a. `cd` *`racine_was`*`/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear`
   `racine_was``/session/samples/splicer.properties`

Le filtre de servlet qui est raccordé conserve les valeurs de configuration par défaut. Vous pouvez remplacer ces valeurs par défaut par des options de configuration que vous spécifiez dans le fichier de propriétés, dans le second argument. Pour une liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

Vous pouvez modifier et utiliser l'exemple de fichier `splicer.properties` fourni avec l'installation d'eXtreme Scale. Vous pouvez également utiliser le script **`addObjectGridServlets`**, qui insère le gestionnaire de sessions en étendant chaque servlet. Mais le script recommandé est le script **`addObjectGridFilter`**.

- **Raccorder manuellement l'application avec le script de génération Ant**

  WebSphere eXtreme Scale est fourni avec un fichier `build.xml` qui peut être utilisé par Apache Ant, qui est inclus dans le dossier *`racine_was`*`/bin` d'une installation WebSphere Application Server. Vous pouvez modifier ce fichier `build.xml` pour modifier les propriétés de configuration du gestionnaire de sessions. Le nom de ces propriétés de configuration est identique au nom des propriétés contenues dans le fichier `splicer.properties`. Pour modifier le fichier `build.xml`, appelez le processus Ant en exécutant la commande suivante :
  - UNIX `ant.sh`, `ws_ant.sh`
  - Windows `ant.bat`, `ws_ant.bat`

  (UNIX) ou (Windows).
- **Mettre à jour manuellement le descripteur Web**

  Editez le fichier `web.xml` fourni avec l'application Web pour incorporer la déclaration de filtre, son mappage de servlets et les paramètres d'initialisation du contexte de servlet. N'utilisez pas cette méthode car elle présente des risques d'erreurs.

Pour connaître la liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

2. Déployez l'application. Déployez l'application à l'aide de votre procédure normale pour un serveur ou un cluster. Une fois que vous avez déployé l'application, vous pouvez la démarrer.
3. Accédez à l'application. Vous pouvez maintenant accéder à l'application, qui interagit avec le gestionnaire de sessions et WebSphere eXtreme Scale.

Que faire ensuite

Vous pouvez modifier la plupart des attributs de configuration du gestionnaire de sessions lorsque vous instrumentez votre application pour utiliser le gestionnaire de sessions. Ces attributs sont : réplication synchrone ou asynchrone, taille de la table de session en mémoire, etc. En dehors des attributs modifiables lors de l'instrumentation de l'application, les seuls autres attributs de configuration que vous pouvez modifier après le déploiement de l'application sont ceux liés à la topologie des clusters de serveurs WebSphere eXtreme Scale et à la manière dont leurs clients (gestionnaires de sessions) s'y connectent.

Comportement de scénarios distant : si la grille de données complète qui héberge les données de sessions d'application est inaccessible à partir du client du conteneur Web, le client utilise le conteneur Web de base dans WebSphere Application Server pour la gestion des sessions. La grille de données peut être

inaccessible dans les cas suivants :

- Problème de réseau entre le conteneur Web et les serveurs de conteneur distants.
- Arrêt des processus serveur de conteneur distant.

Le nombre de références de session conservées en mémoire, spécifié par le paramètre **sessionTableSize**, est maintenu même lorsque les sessions sont stockées dans le conteneur Web de base. Les sessions les moins utilisées sont invalidées à partir du cache de session du conteneur Web lorsque la valeur **sessionTableSize** est dépassée. Si la grille de données distante devient disponible, les sessions qui ont été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données distante n'est pas disponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données distante de production lorsque le système est chargé.

Rubrique parent : Création de grilles de données de session

Java

Configuration du gestionnaire de sessions HTTP avec WebSphere Portal

Vous pouvez rendre persistantes des sessions HTTP à partir de WebSphere Portal dans une grille de données.

Avant de commencer

Votre environnement WebSphere eXtreme Scale Client et WebSphere Portal doivent satisfaire aux exigences suivantes :

- Installez WebSphere eXtreme Scale Client sur vos noeuds WebSphere Application Server et WebSphere Portal. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
- WebSphere Portal version 7 ou suivante.
- Les portlets personnalisés doivent être configurés dans WebSphere Portal. Les portlets d'administration livrés avec WebSphere Portal ne peuvent actuellement pas être intégrés avec des grilles de données.

Pourquoi et quand exécuter cette tâche

L'introduction de WebSphere DataPower XC10 Appliance dans un environnement WebSphere Portal peut être bénéfique dans les cas suivants :

Important : Bien que les scénarios suivants présentent des avantages, l'introduction de WebSphere DataPower XC10 Appliance dans l'environnement peut entraîner une utilisation plus importante des processeurs au niveau de WebSphere.

- **Lorsque la persistance des sessions est requise.**

  Par exemple, si les données de session de vos portlets personnalisés doivent rester disponibles lors d'une défaillance de WebSphere Portal Server, vous pouvez rendre persistantes les sessions HTTP dans la grille de données WebSphere DataPower XC10 Appliance. Les données sont répliquées entre de nombreux serveurs, accroissant la disponibilité des données.

- **Dans une topologie avec plusieurs centres de données.**

  Si votre topologie couvre plusieurs centres de données à travers différents emplacements physiques, vous pouvez rendre persistantes les sessions HTTP de WebSphere Portal dans la grille de données WebSphere DataPower XC10 Appliance. Les sessions sont répliquées dans les grilles de données des centres de données. Si un centre de données est défaillant, les sessions sont basculées vers un autre centre de données qui contient une copie des données de la grille de données.

- **Pour diminuer la mémoire requise au niveau de WebSphere Portal Server.**

  En déchargeant les données de session sur un groupe de serveurs de conteneurs, un sous-ensemble des sessions se trouve sur les serveurs WebSphere Portal. Ce déchargement de données réduit la mémoire requise au niveau de WebSphere Portal Server.

Procédure

1. Configurez l'application WebSphere Portal `wps` et les éventuels portlets personnalisés pour permettre aux sessions d'être stockées dans la grille de données.

   Pour plus d'informations, voir Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données. Cet action aboutit au raccordement des portlets personnalisés pour permettre la persistance de session sur votre grille de données.

2. Si TLS/SSL (Transport Layer Security/Secure Sockets Layer) est configuré pour le serveur WebSphere Portal et pour le dispositif, vous devez configurer les fichiers de clés certifiées TLS/SSL.
   - Si la communication sortante résultante du serveur WebSphere Portal vers le dispositif utilise TLS/SSL, vous devez ajouter le certificat du dispositif à la configuration de WebSphere Application Server. Utilisez le script **`addXC10PublicCert.py`**. Il se trouve dans le répertoire *`racine_was`*`/bin` :

     ```
     wsadmin.bat -conntype SOAP -port <PORT_SOAP_PORTAL_SERVER> -lang jython -user
     wpsadmin -password wpsadmin -f addXC10PublicCert.py
     ```

   - Si la communication entrante résultante du dispositif vers le serveur WebSphere Portal utilise TLS/SSL, mettez à jour le fichier de clés certifiées du dispositif pour inclure les certificats publics pour le serveur WebSphere Portal. La mise à jour du fichier de clés certifiées permet la communication entre le dispositif et WebSphere Portal.

a. Extrayez la clé publique du certificat personnel Portal Server. Utilisez l'utilitaire **IKEYMAN**. Cet utilitaire crée un fichier `.arm`. Pour plus d'informations, voir Extraction de certificats publics pour les fichiers de clés certifiées.
b. Téléchargez le fichier de clés certifiées public pour le dispositif. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..
c. Utilisez l'utilitaire **iKeyman** pour mettre à jour le fichier `truststore.jks` que vous avez extrait du dispositif avec le certificat Portal Server public dans le fichier `.arm`. Pour plus d'informations, voir Importation de certificats de signataire.
d. Téléchargez le fichier de clés certifiées mis à jour vers le dispositif. Cliquez sur **Soumettre les paramètres TLS** après avoir téléchargé le fichier de clés certifiées. La collectivité redémarre automatiquement lorsque vous soumettez les paramètres TLS et le nouveau fichier de clés certifiées est ajouté aux autres dispositifs de la collectivité. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

3. Certaines versions du serveur WebSphere Portal peuvent présenter des erreurs d'exécution lorsque des cookies sont ajoutés à une réponse HTTP. Puisque WebSphere DataPower XC10 Appliance ajoute des cookies pour permettre le basculement et à d'autres fins, ces cookies doivent être ajoutés à la liste des cookies à ignorer du serveur WebSphere Portal. Pour plus d'informations, voir la section décrivant le paramètre cookie.ignore.regex de la rubrique consacrée à la mise en cache des pages partagées par plusieurs utilisateurs, sur le wiki dédié à IBM WebSphere Portal. Les deux cookies qui doivent être ajoutés à la liste sont IBMID.* et IBMSessionHandle.*. La liste mise à jour peut ressembler à ceci : `"digest\\.ignore.*|LtpaToken|LtpaToken2|JSESSIONID|IBMID.*|IBMSessionHandle.*"`. Pour plus d'informations, voir la rubrique consacrée à la mise en cache des pages partagées par plusieurs utilisateurs sur le wiki dédié à IBM WebSphere Portal.
4. Redémarrez les serveurs WebSphere Portal. Pour plus d'informations, voir WebSphere Portal version 7 : Démarrage et arrêt des serveurs, des gestionnaires de déploiement et des agents de noeud.

Résultats

Vous pouvez accéder à WebSphere Portal Server ; les données de session HTTP pour les portlets personnalisés configurés sont conservées dans la grille de données.

Si l'ensemble de la grille de données qui héberge les données de sessions d'application est inaccessible à partir du client de conteneur Web, le client utilise le conteneur Web de base dans la gestion de sessions WebSphere Application Server. La grille de données peut être inaccessible dans les cas suivants :

- Problème de réseau entre le conteneur Web et les serveurs de conteneur distants.
- Arrêt des processus serveur de conteneur distant.

Le nombre de références de session conservées en mémoire, spécifié par le paramètre **sessionTableSize**, est maintenu même lorsque les sessions sont stockées dans le conteneur Web de base. Les sessions les moins utilisées sont invalidées à partir du cache de session du conteneur Web lorsque la valeur **sessionTableSize** est dépassée. Si la grille de données distante devient disponible, les sessions ayant été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données distante n'est pas disponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données distante de production lorsque le système est chargé.

Rubrique parent : Création de grilles de données de session

Tâches associées:
Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications
Administration à l'aide de l'interface de commande HTTP

Information associée:
Gestion des clés avec l'interface graphique IKEYMAN

Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications

WebSphere eXtreme Scale Client est fourni avec une implémentation de gestion de session qui remplace le gestionnaire de sessions par défaut pour un conteneur Web. Cette implémentation offre la réplication de session, la haute disponibilité, une meilleure évolutivité et des options de configuration. Vous pouvez activer le gestionnaire de réplication de session WebSphere eXtreme Scale Client et le démarrage du conteneur intégré ObjectGrid générique pour différents serveurs d'application, tels que Tomcat.

Avant de commencer

Vous pouvez activer Transport Layer Security (TLS) avec SSL sur WebSphere DataPower XC10 Appliance. Pour plus d'informations, voir Configuration de TLS pour les applications de grille de données.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser le gestionnaire de sessions HTTP avec d'autres serveurs d'applications qui n'exécutent pas WebSphere Application Server, tels que WebSphere Application Server Community Edition. Pour configurer d'autres serveurs d'applications pour qu'ils utilisent la grille de données, vous devez raccorder votre application et incorporer des fichiers WebSphere eXtreme Scale Client Java archive (JAR) dans votre application.

Procédure

1. Raccordez votre application de sorte qu'elle puisse utiliser le gestionnaire de sessions. Pour utiliser le gestionnaire de sessions, vous devez ajouter les déclarations de filtre appropriées aux descripteurs de déploiement Web de l'application. En outre, les paramètres de configuration du gestionnaire de sessions sont transmis au gestionnaire de sessions sous la forme de paramètres d'initialisation du contexte de servlet dans les descripteurs de déploiement. Vous disposez de trois manières de présenter ces informations dans votre application :
   - Script **`addObjectGridFilter`** :

     Utilisez un script de ligne de commande fourni avec WebSphere eXtreme Scale Client pour raccorder une application avec des déclarations de filtre et une configuration sous forme de paramètres d'initialisation de contexte de servlet. Le script *rép_base_wxs*`/session/bin/addObjectGridFilter.sh|bat` accepte deux paramètres : le chemin absolu d'accès au fichier EAR (enterprise archive) ou au fichier WAR (web archive) à raccorder et le chemin absolu au fichier des propriétés splicer qui contient diverses propriétés de configuration. La syntaxe de ce script est la suivante :

     Windows

     ```
     addObjectGridFilter.bat <fichier_ear_ou_war> <fichier_propriétés_splicer>
     ```

     UNIX

     ```
     addObjectGridFilter.sh
     <fichier_ear_ou_war> <fichier_propriétés_splicer>
     ```

     UNIX Exemple d'utilisation de WebSphere eXtreme Scale Client installé dans un répertoire autonome sur UNIX :

     a. `cd` *rép_base_wxs*`/session/bin`
     b. `addObjectGridFilter.sh /tmp/mySessionTest.ear` *rép_base_wxs*`/session/samples/splicer.properties`

     Le filtre de servlet qui est joint conserve les valeurs de configuration par défaut. Vous pouvez remplacer ces valeurs par défaut par des options de configuration que vous spécifiez dans le fichier de propriétés, dans le second argument. Pour connaître la liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

     Vous pouvez modifier et utiliser l'exemple de fichier `splicer.properties` fourni avec l'installation d'WebSphere eXtreme Scale Client. Toutefois, vous devez examiner le fichier afin de vous assurer qu'aucune des propriétés que vous voulez utiliser ne figure en commentaire (comme, par exemple, la propriété `catalogHostPort`). Vous pouvez également utiliser le script **`addObjectGridServlets`**, qui insère le gestionnaire de sessions en étendant chaque servlet. Cependant, le script recommandé est le script **`addObjectGridFilter`**.

   - Script de génération Ant :

     WebSphere eXtreme Scale Client est fourni avec un fichier `build.xml` qui peut être utilisé par

Apache Ant, qui est inclus dans le dossier *racine_was*/bin d'une installation WebSphere Application Server. Vous pouvez modifier le fichier build.xml pour changer les propriétés de configuration du gestionnaire de sessions. Les propriétés de configuration correspondent aux noms de propriété dans le fichier splicer.properties. Une fois que le fichier build.xml a été modifié, appelez le processus Ant en exécutant ant.sh, ws_ant.sh (UNIX) ou ant.bat, ws_ant.bat (Windows).

- Mise à jour manuelle du descripteur Web :

  Editez le fichier web.xml qui est packagé avec l'application Web pour incorporer la déclaration de filtre, son mappage de servlets et les paramètres d'initialisation du contexte de servlet. N'utilisez pas cette méthode car elle est source d'erreurs possibles.

Pour connaître la liste des paramètres que vous pouvez utiliser, voir Paramètres d'initialisation du contexte de servlet.

2. Incorporez dans votre application les fichiers JAR du gestionnaire de réplication de sessions d'WebSphere eXtreme Scale Client. Vous pouvez incorporer les fichiers dans le répertoire WEB-INF/lib des modules d'application ou dans le chemin d'accès aux classes du serveur d'applications. Les fichiers JAR requis varient selon le type de conteneurs utilisés :
   - Serveurs de conteneur distants : ogclient.jar et sessionobjectgrid.jar
   - Serveurs de conteneur intégrés : objectgrid.jar et sessionobjectgrid.jar
3. Déployez l'application. Déployez l' application à l'aide de votre procédure normale pour un serveur ou un cluster. Une fois l'application déployée, vous pouvez la démarrer.
4. Accédez à l'application. Vous pouvez maintenant accéder à l'application, qui interagit avec le gestionnaire de sessions et WebSphere eXtreme Scale Client.

Que faire ensuite

Vous pouvez modifier la majorité des attributs de configuration du gestionnaire de sessions lorsque vous instrumentez votre application pour utiliser le gestionnaire de sessions. Ces attributs sont des variantes du type de réplication (synchrone ou asynchrone), la taille de la table des sessions en mémoire, etc. En dehors des attributs modifiables lors de l'instrumentation de l'application, les seuls autres attributs de configuration que vous pouvez modifier après le déploiement de l'application sont ceux liés à la topologie des clusters de serveurs WebSphere eXtreme Scale et à la manière dont leurs clients (gestionnaires de sessions) s'y connectent.

Comportement dans le scénario distant : si l'ensemble de la grille de données qui héberge les données de session d'application est inaccessible depuis le client du conteneur Web, le client utilise à la place le conteneur Web de base du serveur d'applications pour gérer les sessions. La grille de données peut être inaccessible dans les scénarios suivants :

- Problème de réseau entre le conteneur Web et les serveurs de conteneur distants.
- Arrêt des processus serveur de conteneur distant.

Le nombre de références de session conservées en mémoire, spécifié par le paramètre **sessionTableSize**, est maintenu même lorsque les sessions sont stockées dans le conteneur Web de base. Les sessions les moins utilisées sont invalidées à partir du cache de session du conteneur Web lorsque la valeur **sessionTableSize** est dépassée. Si la grille de données distante devient disponible, les sessions ayant été invalidées à partir du cache du conteneur Web peuvent extraire les données de la grille de données distante et charger les données dans une nouvelle session. Si l'ensemble de la grille de données distante n'est pas disponible et que la session est invalidée dans le cache de session, les données de session utilisateur sont perdues. Compte tenu de ce problème, n'arrêtez pas l'ensemble de la grille de données distante de production lorsque le système est chargé.

Rubrique parent : Création de grilles de données de session

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Tâches associées:
Configuration du gestionnaire de sessions HTTP avec WebSphere Portal
Administration à l'aide de l'interface de commande HTTP

Référence associée:
Fichier splicer.properties

Information associée:
Installation de fichiers d'application d'entreprise à l'aide de la console
Installation d'applications d'entreprise à l'aide d'un script wsadmin
Gestion des clés avec l'interface graphique IKEYMAN

.NET

Configuration d'un fournisseur de stockage d'état de session ASP.NET

2.5+ Vous pouvez stocker l'état de session de vos applications ASP.NET dans une grille de données.

Avant de commencer

- Installez WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Présentation de l'installation de WebSphere eXtreme Scale Client for .NET.
- Votre environnement .NET doit respecter la configuration système requise. Pour plus d'informations, voir Remarques relatives à Microsoft .NET.
- Votre application .NET doit être configurée pour gérer l'état de session.

.NET 2.5+ **Création d'une grille de données à utiliser avec le fournisseur de stockage d'état de session ASP.NET**
Créez une grille de données pour enregistrer l'état de session de vos applications ASP.NET.

.NET 2.5+ **Configuration de votre application .NET application pour l'utilisation du fournisseur de stockage d'état de session ASP.NET**
Pour configurer le fournisseur de stockage d'état de session ASP.NET, vous devez mettre à jour le fichier `web.config` de l'application ASP.NET en y définissant le fournisseur de stockage d'état de session ASP.NET et sa configuration.

Rubrique parent : Création de grilles de données de session

Création d'une grille de données à utiliser avec le fournisseur de stockage d'état de session ASP.NET

2.5+ Créez une grille de données pour enregistrer l'état de session de vos applications ASP.NET.

Procédure

Créez une grille de données pour votre fournisseur de stockage d'état de session. Dans l'interface utilisateur, cliquez sur **Grille de données** > **Session**. Cliquez sur l'icône Ajouter ( ) et indiquez le nom de la grille de données de session à créer. Par défaut, le fournisseur de stockage d'état de session ASP.NET est configuré pour utiliser une grille nommée `session`. Si vous créez une grille avec un nom différent, configurez le fournisseur avec ce nom.

Que faire ensuite

Mettez à jour la configuration de votre application Web afin que cette dernière utilise la grille de données. Pour plus d'informations, voir Configuration de votre application .NET application pour l'utilisation du fournisseur de stockage d'état de session ASP.NET.

Rubrique parent : .NET 2.5+ Configuration d'un fournisseur de stockage d'état de session ASP.NET

.NET

Configuration de votre application .NET application pour l'utilisation du fournisseur de stockage d'état de session ASP.NET

2.5+ Pour configurer le fournisseur de stockage d'état de session ASP.NET, vous devez mettre à jour le fichier `web.config` de l'application ASP.NET en y définissant le fournisseur de stockage d'état de session ASP.NET et sa configuration.

Avant de commencer

- Configurez une grille de données pour le stockage de l'état de session HTTP ASP.NET. Pour plus d'informations, voir Création d'une grille de données à utiliser avec le fournisseur de stockage d'état de session ASP.NET.
- Vous devez connaître l'hôte et le port de serveur de catalogue. Pour obtenir l'hôte et le port du serveur de catalogue dans l'interface utilisateur du dispositif, cliquez sur **Collectivité** > **Membres**. Sélectionnez un membre de collectivité. L'adresse IP et le numéro de port du serveur de catalogue s'affichent.

Procédure

1. Mettez à jour le fichier `web.config` de votre application ASP.NET en y définissant les paramètres du fournisseur de stockage d'état de session ASP.NET. Vous devez mettre à jour ou ajouter dans le fichier `web.config` le texte qui figure en gras dans l'exemple suivant :

```
<system.web>
        …
        <sessionState
                mode="Custom"
                customProvider="WxsSessionStateStoreProvider">
                <providers>
                        <add
                                name="WxsSessionStateStoreProvider"
type="IBM.WebSphere.Caching.SessionStateStore.WxsSessionStateStore,
                                        IBM.WebSphere.Caching,
Version=8.6.0.2, Culture=neutral,
                                        PublicKeyToken=b439a24ee43b0816"
                                wxsPropertyFile="\optional\path\to\NET-
client.properties"
                                wxsHostAndPort="optionalHostAndPort"
                                wxsGridName="session"
                                wxsMapName="ASPNET.SessionState"

         />
        </providers>
                </sessionState>
        …
</system.web>
```

wxsPropertyFile (facultatif)

Indique le nom entièrement qualifié du fichier de propriétés que le fournisseur utilise lorsqu'il se connecte à la grille de données via l'API Connect. Si cet attribut n'est pas spécifié ou contient une chaîne vide, le fournisseur recherche le fichier `Client.Net.properties` dans le répertoire d'exécution en cours du processus de l'application Web. Si le fournisseur ne trouve pas ce fichier dans ce répertoire, il le recherche dans le répertoire *net_client_home*`\config`.

wxsHostAndPort

Indique la liste séparée par des virgules des paires hôte et port correspondant aux serveurs de catalogue auxquels le fournisseur de stockage d'état de session se connecte lorsqu'il accède à la grille de données. Le format est le suivant :

```
<nom ou adresse IP de l'hôte>:<port tcp>[,<nom ou adresse IP de l'hôte>:<port tcp>]
```

wxsGridName (facultatif)

Indique le nom de la grille de données à laquelle le fournisseur de stockage de session ASP.NET se connecte. Si vous avez créé une grille de données pour les états de session ASP.NET, indiquez son nom. Si vous n'indiquez pas de valeur, le fournisseur se connecte à la grille de données `session`.

wxsMapName (facultatif)

Indique la mappe à laquelle le fournisseur se connecte. Si vous n'indiquez pas de valeur, le fournisseur se connecte à la mappe `ASPNET.SessionState`.

2. Redémarrez l'application Web cible. L'application Web doit redémarrer afin qu'IIS puisse charger le fournisseur. Dans la plupart des cas, une fois le fichier `web.config` modifié et le traitement de la demande HTTP en cours terminé, le redémarrage s'effectue automatiquement.

Résultats

L'état de session ASP.NET de votre application ASP.NET est stocké dans la grille de données.

Rubrique parent : .NET 2.5+ Configuration d'un fournisseur de stockage d'état de session ASP.NET

Java

Affichage de la taille des sessions HTTP

2.5+ Vous pouvez utiliser l'utilitaire **xscmd** pour afficher la taille des sessions dans votre application Java. Cette information peut vous aider à identifier et résoudre les problèmes qui peuvent survenir dans votre grille de données.

Avant de commencer

- L'utilitaire **xscmd** doit être configuré pour se connecter à vos serveurs de catalogue. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.
- Vous devez connaître l'ID et le descripteur de la session dont vous voulez afficher la taille.

Procédure

Exécutez la commande **xscmd -c showSessionSize**. La syntaxe de cette commande est la suivante :

```
xscmd.bat|sh -c showSessionSize   -cep nom_hôte:port(,nom_hôte:port)
-sid ID_session -sh descripteur_session -g nom_grille
[-wacr racine_contexte_application_web]
```

Par exemple, vous pouvez exécuter la commande suivante :

```
xscmd.bat -c showSessionSize -g session -cep 9.42.139.213:2809 -sh c -sid
LFMzQnDX5k87xztMF3ri6jU -wacr /A -user xcadmin -pwd xcadmin
```

Cette commande affiche les informations de session suivantes :

```
*** ID session : LFMzQnDX5k87xztMF3ri6jU
  Nombre d'attributs de session : 2
  Taille totale des métadonnées de session : 488 octets
  Taille totale des attributs de session : 243 octets
  Taille totale de session : 731 octets

  Clé et valeur de métadonnées :
  Clé : LFMzQnDX5k87xztMF3ri6jU/A
  Taille de la clé : 88 octets
  Taille de la valeur : 400 octets

  Clé et valeur d'attribut individuel :
  Clé : LFMzQnDX5k87xztMF3ri6jU/A_session.reqCount
  Taille de la clé : 128 octets
  Taille de la valeur : 2 octets
  Clé : LFMzQnDX5k87xztMF3ri6jU/A_user
  Taille de la clé : 104 octets
  Taille de la valeur : 9 octets
```

Rubrique parent : Création de grilles de données de session

Paramètres d'initialisation du contexte de servlet

La liste de paramètres d'initialisation du contexte de servlet suivante peut être spécifiée dans le fichier splicer.properties en fonction de la méthode de raccord choisie.

Paramètres

`applicationQualifiedCookies`

Valeur de type chaîne `true` ou `false`. Utilisez la valeur `true` si votre environnement contient plusieurs applications qui utilisent des noms de cookie uniques. La valeur par défaut est `false`, ce qui suppose que toutes les applications utilisent le même nom de cookie.

`authenticationRetryCount`

Indique le nombre de tentatives d'authentification qui ont lieu lorsque les données d'identification ont expiré. Si la valeur est 0, aucune nouvelle tentative d'authentification n'a lieu.

`catalogHostPort`

Le serveur de catalogues peut être contacté pour obtenir une instance ObjectGrid côté client. La valeur doit avoir le format `hôte:port<,hôte:port>`. L'hôte est l'hôte d'écoute sur lequel le serveur de catalogue s'exécute. Le port est le port d'écoute du processus serveur de catalogue. Cette liste peut avoir une longueur quelconque et n'est utilisée que pour l'amorçage. La première adresse viable est utilisée. Elle est facultative dans WebSphere Application Server si la propriété **`catalog.services.cluster`** est définie.

`credentialAuthentication`

Indique la prise en charge de l'authentification des données d'identification du client. Les valeurs possibles sont :

- Jamais : le client ne prend pas en charge l'authentification des données d'identification.
- Pris en charge : le client prend en charge l'authentification des données d'identification si et seulement si le serveur en fait de même.
- Obligatoire : le client requiert l'authentification des données d'identification. La valeur par défaut est Pris en charge.

`cookieDomain`

Spécifie si vous exigez que les sessions soient accessibles à travers les hôtes. Définissez la valeur avec le nom du domaine commun entre les hôtes.

`cookiePath`

Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe est utilisée pour obtenir les données d'identification des clients.

`credentialGeneratorClass`

Le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe sert à obtenir les données d'identification des clients.

`credentialGeneratorProps`

Les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés sont affectées à l'objet à l'aide de la méthode setProperties(String). La valeur credentialGeneratorProps n'est utilisée que si la valeur de la propriété **`credentialGeneratorClass`** n'est pas NULL.

`enableSessionStats`

Valeur de type chaîne `true` ou `false`. Active le suivi des statistiques des sessions HTTP client eXtreme Scale.

`fragmentedSession`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `true`. Ce paramètre permet de contrôler si le produit stocke les données de session en tant qu'entrée d'un seul bloc ou s'il stocke chaque attribut séparément.

Utilisez la valeur `true` si la session d'application Web a de nombreux attributs ou des attributs de grande taille. Utilisez la valeur `false` si une session a peu d'attributs, car dans ce cas tous les attributs sont stockés dans la même clé dans la grille de données.

Dans la précédente implémentation à base de filtres, il était fait référence à cette propriété en tant que mécanisme de persistance avec, comme valeurs possibles, ObjectGridStore (fragmentation) et ObjectGridAtomicSessionStore (non-fragmentation).

objectGridType

Valeur de type chaîne `REMOTE` ou `EMBEDDED`. La valeur par défaut est `REMOTE`.

Si la valeur est `REMOTE`, les données de session sont stockées en dehors du serveur sur lequel l'application Web est exécutée.

Si la valeur est `EMBEDDED`, un conteneur intégré eXtreme Scale démarre dans le processus serveur d'applications sur lequel l'application Web s'exécute.

objectGridName

Valeur de chaîne qui définit le nom de l'instance ObjectGrid utilisée pour une application Web particulière. Le nom par défaut est session.

Cette propriété doit refléter le nom objectGridName dans les fichiers XML ObjectGrid et XLM de déploiement utilisés pour démarrer les serveurs de conteneur eXtreme Scale.

objectGridXML

L'emplacement du fichier `objectgrid.xml`. Le fichier XML intégré regroupé dans la bibliothèque eXtreme Scale est chargé automatiquement si `objectGridType=EMBEDDED` et que la propriété **objectGridXML** n'est pas définie.

objectGridDeploymentXML

Indique l'emplacement du fichier XML de stratégie de déploiement d'objectGrid. Le fichier XML intégré regroupé dans la bibliothèque eXtreme Scale est chargé automatiquement si `objectGridType=EMBEDDED` et que la propriété **objectGridDeploymentXML** n'est pas définie.

replicationInterval

Entier (en secondes) qui définit le temps séparant deux écritures de sessions actualisées vers la grille. La valeur par défaut est `10` secondes. Les valeurs possibles sont comprises entre 0 et 60. 0 signifie que les sessions actualisées sont écrites dans la grille pour chaque demande dès la fin de l'appel à la méthode de service du servlet. Une valeur **replicationInterval** plus élevée améliore les performances, car un moins grand nombre de mises à jour sont écrites dans la grille de données. Mais en même temps, une valeur supérieure à 0 rend la configuration moins tolérante aux pannes.

Ce paramètre ne s'applique que lorsque objectGridType a la valeur `REMOTE`.

reuseSessionID

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. A la valeur `true` si le conteneur Web sous-jacent réutilise les ID de session dans les requêtes aux différents hôtes. La valeur de cette propriété doit être la même que la valeur du conteneur Web. Si vous utilisez WebSphere Application Server et configurez la persistance de session HTTP eXtreme Scale en utilisant la console d'administration ou le scriptage de l'outil **wsadmin**, la propriété personnalisée du conteneur Web `HttpSessionIdReuse=true` est ajoutée par défaut. **reuseSessionID** a également la valeur `true`. Si vous ne voulez pas réutiliser l'ID de session, définissez la propriété `HttpSessionIdReuse=false` dans la propriété personnalisée du conteneur Web avant de configurer la persistance de session eXtreme Scale.

sessionIdOverrideClass

Nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.xs.sessionmanager.SessionIDOverride. Cette classe est utilisée pour remplacer l'identificateur de session unique obtenu avec la méthode HttpSession.getId() afin que toutes les applications aient le même ID. Par défaut, l'ID provenant de HttpSession.getId() est utilisé.

sessionStatsSpec = session.all = enabled

Chaîne de spécification des statistiques HTTP client eXtreme Scale.

shareSessionsAcrossWebApps

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Spécifie si les sessions sont partagées entre des applications Web ; indiquez la valeur de chaîne `true` ou `false`. La spécification de servlet stipule que les sessions HTTP ne peuvent pas être partagées entre des applications Web. Une extension à la spécification de servlet est fournie pour permettre ce partage.

sessionTableSize

Entier qui définit le nombre de références de session conservées en mémoire. La valeur par défaut est `1000`.

Ce paramètre appartient uniquement à une topologie `REMOTE`, car la topologie `EMBEDDED` a déjà les données de session dans le même groupe que le conteneur Web.

Les sessions sont expulsées de la table interne en fonction de la logique LRU (least recently used). Lorsqu'une session est expulsée de cette table, elle est invalidée dans le conteneur Web. Cependant, les données ne sont pas pour autant supprimées de la grille, ce qui permet aux demandes ultérieures de cette session de continuer à extraire les données. Cette valeur doit être supérieure à la valeur maximale du pool d'unités d'exécution du conteneur Web, ce qui réduit les conflits au niveau du cache de session.

`securityEnabled`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Ce paramètre active la sécurité du client eXtreme Scale. Il doit correspondre au paramètre **`securityEnabled`** dans le fichier des propriétés sur serveur eXtreme Scale. Si les paramètres ne correspondent pas, une exception est générée.

`sessionIdOverrideClass`

Remplace l'ID session extrait d'une application. L'ID provenant de la méthode HttpSession.getId() est utilisé par défaut. Permet aux sessions HTTP client eXtreme Scale de remplacer l'ID de session unique d'une application afin que toutes les applications soient récupérées avec le même ID. Défini sur l'implémentation de l'interface com.ibm.websphere.xs.sessionmanager.SessionIDOverride. Cette interface détermine l'ID HttpSession d'après l'objet HttpServletRequest.

`traceSpec`

Spécifie la spécification de trace d'IBM® WebSphere comme une valeur de chaîne. Utilisez ce paramètre pour des serveurs d'applications autres que WebSphere Application Server.

`traceFile`

Spécifie l'emplacement du fichier de trace sous forme de valeur de chaîne. Utilisez ce paramètre pour des serveurs d'applications autres que WebSphere Application Server.

`useURLEncoding`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Affectez-lui la valeur `true` pour activer la réécriture d'URL. La valeur par défaut est `false`, ce qui indique que les cookies sont utilisés pour stocker les données de session. La valeur de ce paramètre doit être identique à celle des paramètres de conteneur Web pour la gestion des sessions.

`useCookies`

Valeur de type chaîne `true` ou `false`. A la valeur true si le conteneur Web sous-jacent réutilise les ID de session dans les requêtes aux différents hôtes. La valeur par défaut est `false`. La valeur de cette propriété doit être la même que la valeur définie dans le conteneur Web.

Rubrique parent : Création de grilles de données de session

Fichier `splicer.properties`

Le fichier `splicer.properties` contient toutes les options de configuration qui permettent de configurer un gestionnaire de sessions basé sur un filtre de servlet.

Exemple de fichier splicer.properties

Si vous décidez d'utiliser l'une des propriétés supplémentaires décrites dans ce fichier, veillez à supprimer la mise en commentaire des lignes des propriétés à activer.

```
# Fichier de propriétés qui contient toutes les options de configuration
# que le gestionnaire de sessions ObjectGrid basé sur un filtre de servlet peut être
configuré pour utiliser.
#
# Ce fichier de propriétés peut être généré pour attribuer toutes les
# valeurs par défaut à ces paramètres de configuration, et permettre de
# remplacer les paramètres individuels à l'aide des propriétés de tâche ANT,
# si ce fichier de propriétés est utilisé avec
# la tâche ANT filtersplicer.

# Valeur de chaîne "REMOTE" ou "EMBEDDED". La valeur par défaut est REMOTE.
# Si elle est définie sur "REMOTE", les données de session seront stockées en dehors du
# serveur où est exécutée l'application Web. Si sa valeur est
# "EMBEDDED", un conteneur WebSphere eXtreme Scale imbriquée démarre
# dans le processus de serveur d'applications dans lequel l'application Web est exécutée.

objectGridType = REMOTE

# Valeur de chaîne qui définit le nom de l'instance ObjectGrid
# utilisée pour une applications Web donnée. Le nom par défaut
# est session. Cette propriété doit refléter l'objectGridName dans les deux
# fichiers xml objectgrid et de déploiement utilisés pour démarrer les conteneurs eXtreme
# Scale.

objectGridName = session

# Le serveur de catalogues peut être contacté pour obtenir une instance
# ObjectGrid côté client. La valeur doit avoir le format
# "host:port<,hôte:port>", où hôte représente l'hôte d'écoute
# sur lequel le serveur de catalogue est en cours d'exécution, et port représente
# le port d'écoute du processus du serveur de catalogue. Cette liste
# peut être d'une longueur quelconque et n'est utilisée que pour l'amorçage.
# La première adresse valide est utilisée. Elle est facultative dans WebSphere
# si la propriété catalog.services.cluster est définie.

# catalogHostPort = host:port<,hôte:port>

# Entier (secondes) qui définit la durée en secondes entre
# l'écriture de sessions actualisées dans ObjectGrid. La valeur par défaut est 10. Cette
propriété
# est utilisée uniquement lorsque objectGridType a la valeur REMOTE. Les valeurs possibles
sont
# comprises entre 0 et 60. 0 signifie que les sessions actualisées sont écrites dans
l'ObjectGrid
# à la fin de l'appel à la méthode de service de servlet de chaque demande.

replicationInterval = 10

# Entier qui définit le nombre de références de session
# conservées en mémoire. La valeur par défaut est 1 000. Cette propriété n'est utilisée
# que lorsqueobjectGridType a la valeur REMOTE. Lorsque le nombre de sessions stockées
# dans la mémoire dans le conteneur Web dépasse cette valeur, la première session ayant
# fait l'objet d'un accès est invalidée depuis le conteneur Web. Si une demande
# arrive pour cette session une fois qu'elle a été invalidée, une nouvelle session
# est créée (avec un nouvel ID de session reuseSessionId=false),
# remplie avec les attributs de la session invalidée. Cette value doit toujours être
# supérieure à la taille maximale du pool d'unités
# d'exécution du conteneur pour éviter les conflits dans ce cache de session.
```

```
sessionTableSize = 1000

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "true".
# Permet de contrôler si nous stockons les données de session comme entrée
# intégrale ou de stocker chaque attribut séparément.
# Cette propriété s'appelle persistenceMechanism dans l'implémentation
# basée sur un filtre précédente, avec les valeurs possibles
# ObjectGridStore (fragmenté) et ObjectGridAtomicSessionStore
# (non fragmenté).

fragmentedSession = true

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "false".
# Active la sécurité du client eXtreme Scale. Ce paramètre doit correspondre
# au paramètre securityEnabled dans le fichier des propriétés du serveur eXtreme
# Scale. Si les paramètres ne correspondent pas, une exception est générée.

securityEnabled = false

# Spécifie la prise en charge de l'authentification des données d'identification du
client.
#   Les valeurs possibles sont les suivantes :
#   Jamais : le client ne prend pas en charge l'authentification des données
d'identification.
#   Pris en charge* : le client prend en charge l'authentification des données
d'identification si et seulement si le serveur
#                     la prend en charge également.
#   Requise : le client requiert l'authentification des données d'identification.
#   Elle est prise en charge par défaut.

# credentialAuthentication =

# Indique le nombre de tentatives d'authentification si les données d'identification
# ont expiré. Si la valeur est 0, aucune tentative d'authentification
# n'a lieu.

# authenticationRetryCount =

# Indique le nom de la classe qui implémente l'interface
# com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator.
# Cette classe est utilisée pour obtenir les données d'identification des clients.

# credentialGeneratorClass =

# Spécifie les propriétés de la classe d'implémentation
# CredentialGenerator. Les propriétés sont définies dans l'objet avec la méthode
# setProperties(String). La valeur credentialGeneratorProps n'est utilisée que si
# la valeur de la propriété credentialGeneratorClass est null.

# credentialGeneratorProps =

# Emplacement du fichier xml objectgrid.
# Le fichier xml pré-intégré qui est regroupé dans la bibliothèque eXtreme Scale
# sera automatiquement chargé si cette propriété
# n'est pas spécifiée et que objectGridType=EMBEDDED

# objectGridXML =

# Emplacement du fichier xml de stratégie de déploiement objectGrid.
# Le fichier xml pré-intégré qui est regroupé dans la bibliothèque eXtreme Scale
# sera automatiquement chargé si cette propriété
# n'est pas spécifiée et que objectGridType=EMBEDDED

# objectGridDeploymentXML =

# Chaîne de spécification de trace IBM WebShere,
# utile pour tous les autres serveurs d'applications, outre WebSphere.
```

```
# traceSpec =

# Chaîne d'emplacement de fichier de trace. # utile pour tous les autres serveurs
d'applications, outre WebSphere.

# traceFile=

# Cette propriété doit être définie pour que les sessions soient
# accessibles sur les hôtes. La valeur sera le nom du domaine
# commun aux hôtes.

# cookieDomain=

# Cette propriété doit être affectée du chemin que vous avez configuré
# pour les paramètres de cookie de serveur d'applications. Le chemin par défaut
# est /.

# cookiePath

# A la valeur true si le conteneur Web sous-jacent
# réutilise l'ID dans les demandes à différents hôtes. La valeur par défaut
# est false. La valeur doit correspondre à celle définie dans
# le conteneur Web.

# reuseSessionId=

# Une valeur de type chaîne "true" ou "false". La valeur par défaut est
# "false". Conformément à la spécification de servlet, les sessions HTTP
# ne peuvent pas être partagées dans les applications Web. Une extension à la
spécification de servlet
# est fournie pour autoriser le partage.

# shareSessionsAcrossWebApps = false

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "false". # Affectez-
lui la valeur true si vous voulez activer la réécriture d'URL (urlRewriting). La valeur
par défaut est
# false. La valeur doit être identique à celle définie dans
# les paramètres de conteneur Web de la gestion de session.

# useURLEncoding = false

# False si vous voulez désactiver les cookies en tant que
# mécanisme de suivi. La valeur par défaut est true. La valeur doit être identique à celle
définie dans
# les paramètres de conteneur Web de la gestion de session.

# useCookies = true

# Valeur de type chaîne "true" ou "false". La valeur par défaut est "false".
# Active le suivi des statistiques des sessions HTTP client eXtreme Scale.

# enableSessionStats = false

# Remplace l'ID session extraite d'une application. # L'ID provenant de la méthode
HttpSession.getId() est utilisé par défaut.
# Permet aux sessions HTTP client eXtreme Scale de remplacer
# l'ID session unique d'une application afin que toutes les applications soient extraites
# avec le même ID.
# Défini avec l'implémentation de
# l'interface com.ibm.websphere.xs.sessionmanager.SessionIDOverride.
# Cette interface détermine l'ID HttpSession d'après
# l'objet HttpServletRequest.

# sessionIdOverrideClass =  # Spécification des statistiques HTTP client eXtreme Scale .

# sessionStatsSpec = session.all = enabled
```

```
# True si votre environnement contient plusieurs applications qui
# utilisent des noms de cookie. False, qui suppose que toutes les applications
# utilisent le même nom de cookie.

# applicationQualifiedCookies=false
```

Rubrique parent : Création de grilles de données de session

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Tâches associées:
Migration d'une réplication de mémoire à mémoire ou d'une session de base WebSphere Application Server pour utiliser la gestion de session WebSphere eXtreme Scale
Prise en note des paramètres de configuration précédents dans la console d'administration WebSphere Application Server
Création du domaine de service de catalogue pour la gestion de sessions dans la grille de données
Configuration de WebSphere DataPower XC10 Appliance pour utiliser les paramètres de configuration précédents
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de la persistance des sessions HTTP WebSphere Application Server dans une grille de données
Administration à l'aide de l'interface de commande HTTP
Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications

Information associée:
Installation de fichiers d'application d'entreprise à l'aide de la console
Installation d'applications d'entreprise à l'aide d'un script wsadmin

Création de grilles de données en cache dynamique

IBM® WebSphere DataPower XC10 Appliance permet de stocker des données pour une instance de cache dynamique WebSphere Application Server. En configurant cette fonction, vous permettez aux applications qui utilisent une instance de cache dynamique WebSphere Application Server d'utiliser les fonctions et les fonctionnalités de performance du dispositif.

Avant de commencer

- Vous devez installer WebSphere eXtreme Scale Client dans votre configuration WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client.
- Si la sécurité de la couche de transport du dispositif est activée ou que vous voulez que les clients utilisent la sécurité de la couche de transport, vous devez activer également la sécurité globale dans la console d'administration WebSphere Application Server. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..

Pourquoi et quand exécuter cette tâche

Lorsque vous configurez le fournisseur de mémoire cache dynamique dans WebSphere Application Server pour utiliser DataPower XC10 Appliance, les données de cache sont stockées en dehors de la topologie WebSphere Application Server. Toutes les données de cache sont stockées sur le dispositif. La mémoire utilisée pour la mise en cache de vos serveurs d'applications peut être utilisée à d'autres fins.

Pour savoir comment utiliser des grilles de données de cache dynamique WebSphere eXtreme Scale Client et DataPower XC10 Appliance avec IBM WebSphere Commerce, consultez les rubriques suivantes dans la documentation d'IBM WebSphere Commerce :

- Activation du service de cache dynamique et mise en cache de servlet
- Activation du cache de données WebSphere Commerce

Procédure

1. Créer une instance de mémoire cache à configurer avec DataPower XC10 Appliance. Pour plus d'informations, voir Configuration des instances de cache dynamique..
2. Configurer le service de catalogue. Le service de catalogue permet à la configuration de la mémoire cache dynamique de WebSphere Application Server de communiquer avec DataPower XC10 Appliance. Vous pouvez configurer le service de catalogue dans la console d'administration de WebSphere Application Server en créant un domaine de service de catalogue. Pour plus d'informations, voir Création de domaines de service de catalogue dans WebSphere Application Server.
3. Créez la grille de données sur DataPower XC10 Appliance et configurez les paramètres de sécurité nécessaires. Vous pouvez exécuter le script `dynaCfgToAppliance` ou créer la configuration manuellement.

   Création de la configuration de la grille de données à l'aide du script `dynaCfgToAppliance` :

   Ce script est installé dans le répertoire `bin` du profil du gestionnaire de déploiement lorsque vous installez WebSphere eXtreme Scale Client. Avant d'exécuter ce script, vérifiez que le dispositif et le gestionnaire de déploiement sont en cours d'exécution :

   ```
   dynaCfgToAppliance <adresse_IP> <nom_jndi_cache> <admin_dispositif> <mdp_admin>
   <port_SOAP> <soap.client.props>
   ```

   `Adresse_IP`

   Spécifie l'adresse IP du système DataPower XC10 Appliance sur lequel vous souhaitez stocker vos données de mémoire cache dynamique.

   `nom_jndi_cache`

   Indique le nom du cache dynamique. Si le nom JNDI du cache dynamique contient des barres obliques (`/`), celles-ci sont converties en tirets pour le nom de la grille de données dans DataPower XC10 Appliance. Par exemple, si le nom du cache dynamique est `services/cache1`, la grille de données créée dans le dispositif s'appelle `services-cache1`. Vous ne pouvez pas utiliser les caractères suivants dans le nom de la grille de données dans DataPower XC10 Appliance : `^ . \\ / , # $ @ : ; \ * ? < > | = + & % [ ] " "`.

   `admin_dispositif`

   Spécifie l'ID administrateur à utiliser pour authentification auprès de l'interface utilisateur de DataPower XC10 Appliance.

   `mdp_admin`

   Spécifie le mot de passe administrateur à utiliser pour authentification auprès de l'interface

utilisateur de DataPower XC10 Appliance.

port_SOAP

(Facultatif) Spécifie le port SOAP du gestionnaire de déploiement si vous n'utilisez pas le port par défaut (8879).

soap.client.props

(Facultatif) Définit le chemin d'accès au fichier `soap.client.props`. Vous devez spécifier ce fichier si vous avez activé la sécurité dans WebSphere Application Server. Ce fichier active la sécurité SOAP et définit le nom d'utilisateur et le mot de passe pour administrer le gestionnaire de déploiement WebSphere Application Server :

```
com.ibm.SOAP.securityEnabled=true
com.ibm.SOAP.loginUserid=
com.ibm.SOAP.loginPassword=
```

Voir fichiers de propriétés du connecteur SOAP et du connecteur Inter-Process Communications pour plus d'informations sur le fichier `soap.client.props`.

Ce script crée la grille de données sur le dispositif. Il définit également les données d'identification par ID et mot de passe spécifiques à DataPower XC10 Appliance que vous avez spécifiées avec les paramètres **admin_dispositif** et **mdp_admin** à l'aide des propriétés personnalisées suivantes :

- xc10.<*nom_grille_données*>.userid
- xc10.<*nom_grille_données*>.password

Ces noms de propriétés ne respectent pas la casse des caractères. La valeur du mot de passe est codée. Si vous exécutez de nouveau le script après la configuration initiale, les propriétés personnalisées sont mises à jour.

Création manuelle de la configuration de la grille de données :

a. Créez la grille de données de mémoire cache dynamique dans l'interface graphique de DataPower XC10 Appliance. Cliquez sur **Grille de données** > **Cache dynamique**. Le nom du cache doit correspondre au nom JNDI du cache dynamique dans la configuration WebSphere Application Server. Lorsque vous entrez le nom JNDI, remplacez les barres obliques (`/`) par des tirets pour le nom de la grille de données dans DataPower XC10 Appliance. Par exemple, si le nom du cache dynamique est `services/cache1`, la grille de données créée dans le dispositif doit s'appeler `services-cache1`.
b. Créez les propriétés personnalisées xc10.<*nom_grille_données*>.userid et xc10.<*nom_grille_données*>.password sur la cellule WebSphere Application Server. La valeur de <*nom_grille_données*> dans chaque propriété personnalisée correspond au nom JNDI de la grille de données, les barres obliques (`/`) étant remplacées par des tirets. Par exemple, dans l'exemple précédent, les noms de propriété personnalisée sont `xc10.services-cache1.userid` et `xc10.services-cache1.password`. Les valeurs doivent correspondre à un ID utilisateur et un mot de passe qui peuvent accéder à la grille de données dans la configuration de DataPower XC10 Appliance. Vous pouvez coder le mot de passe à l'aide du script `encodePassword`, qui se trouve dans le répertoire `bin` du gestionnaire de déploiement.

4. 2.5+ Facultatif : Vous pouvez choisir d'activer la réplication multimaître dans le fournisseur de cache dynamique DataPower XC10 Appliance. Pour plus d'informations, voir Configuration d'une réplication multimaître entre les collectivités.

   Remarque : Les utilisateurs de la grille du cache dynamique de WebSphere Portal Server ou de WebSphere Commerce Server peuvent avoir défini plusieurs instances de cache au sein de leur configuration WebSphere Application Server. Si vous décidez d'activer la réplication multimaître pour DataPower XC10 Appliance, cette configuration n'affecte que les instances de cache qui sont définies pour l'utilisation de DataPower XC10 Appliance en tant que fournisseur de mémoire cache dynamique ; elle n'aura aucun effet sur les instances de cache définies pour l'utilisation du fournisseur de mémoire cache dynamique WebSphere Application Server par défaut.

Résultats

La configuration du fournisseur de mémoire cache dynamique avec le dispositif permet de réduire la quantité de mémoire nécessaire aux serveurs d'applications. Toutes les données de mémoire cache sont transférées vers le dispositif et disparaisse de la mémoire des serveurs d'applications.

Que faire ensuite

- Configurez la sécurité avant de commencer à envoyer des données à la grille de données. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
- Configurez des répliques. Les répliques permettent de s'assurer que les données de vos grilles de données sont disponibles si la copie principale échoue. Pour configurer des répliques, cliquez sur

Grille de données > **Mémoire cache dynamique** > **Afficher les attributs avancés**. Les répliques ne sont créées que si le dispositif fait partie d'une collectivité. Si le nombre de dispositifs de la collectivité est *n*, le nombre maximal de répliques est *n - 1*. Ainsi, si vous configurez trois répliques mais que vous n'avez que deux dispositifs dans la collectivité, une seule réplique est créée. Des répliques supplémentaires sont créées si vous ajoutez des dispositifs à la collectivité. Indiquez comme nombre de répliques le nombre idéal de répliques souhaitées : ainsi, lorsque des dispositifs rejoignent la collectivité, de nouvelles répliques sont créées. Le contenu de la grille de données est effacé lorsque vous modifiez le nombre de répliques.

- La limite de capacité de la grille pour une instance de mémoire cache dynamique WebSphere Application Server est configurée lors de la création de cette instance. Si cette taille est dépassée, une stratégie d'éviction LRU (least recently used) est utilisée pour maintenir le nombre d'entrées dans la limite configurée.
- Vous pouvez contrôler la grille de données de mémoire cache dynamique à partir de l'interface utilisateur de DataPower XC10 Appliance. Pour plus d'informations, voir Surveillance des grilles de données dans l'interface utilisateur.

Java 2.5+ **Configuration des instances de cache dynamique**

Le service de mise en mémoire cache de WebSphere Application Server permet la création d'une instance de cache par défaut (baseCache) et d'instances de cache de servlet et d'objet supplémentaires.

Java 2.5+ **Présentation du fournisseur de cache dynamique**

WebSphere Application Server fournit un service de cache dynamique disponible pour déployer des applications Java EE. Ce service permet de mettre des données en cache, telles que la sortie d'un servlet, une page JSP ou des commandes, ainsi que les données d'objet définies par programme dans une application d'entreprise en utilisant des API DistributedMap. En configurant cette fonction, vous pouvez activer les applications qui utilisent le service de cache dynamique pour qu'elles utilisent les fonctions et les fonctionnalités de performance de WebSphere DataPower XC10 Appliance.

Java **Création de domaines de service de catalogue dans WebSphere Application Server**

A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données de cache dynamique.

Java 2.5+ **Configuration d'un cache local pour la mémoire cache dynamique**

Vous pouvez configurer un cache local afin d'utiliser une grille de données de mémoire cache dynamique sur le dispositif ou la collectivité. Le cache local utilise des ressources JVM locales. Généralement, le cache local comporte un sous-ensemble des données qui figurent dans la grille de données de cache dynamique sur le dispositif.

Rubrique parent : Configuration des grilles de données

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Configuration des instances de cache dynamique

Le service de mise en mémoire cache de WebSphere Application Server permet la création d'une instance de cache par défaut (baseCache) et d'instances de cache de servlet et d'objet supplémentaires.

Pourquoi et quand exécuter cette tâche

L'instance de cache par défaut (baseCache) était initialement la seule instance de cache dynamique prise en charge par WebSphere Application Server et elle est actuellement l'instance de cache dynamique standard utilisée par WebSphere Commerce Suite. Les instances supplémentaires de servlet et d'objet ont été ajoutées dans les dernières versions de WebSphere Application Server et sont configurées dans une section **Instance de Cache** distincte de la console d'administration WebSphere Application Server.

Java 2.5+ **Configuration de l'instance de mémoire cache dynamique par défaut (baseCache)**
L'instance de mémoire cache dynamique par défaut est créée par le service de mise en mémoire cache dynamique WebSphere Application Server. Cette instance de mémoire cache dynamique de servlet est utilisée par des produits tels qu'IBM WebSphere Commerce. Contrairement aux autres instances de mémoire cache définies avec WebSphere Application Server, baseCache est propre à un seul serveur ou à une seule instance de cluster. Utilisez cette procédure pour configurer l'instance baseCache dans WebSphere Application Server afin de l'utiliser comme fournisseur de cache dynamique avec WebSphere eXtreme Scale.

Java 2.5+ **Configuration des instances de cache dynamique d'objet ou de servlet**
WebSphere Application Server permet de configurer des instances de cache dynamique d'objet ou de servlet en plus de l'instance par défaut. Pour configurer ces instances supplémentaires, procédez comme suit.

Java 2.5+ **Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées**
WebSphere Application Server permet de définir des propriétés personnalisées dans une instance de cache dynamique.

Rubrique parent : Java Création de grilles de données en cache dynamique

Java

Configuration de l'instance de mémoire cache dynamique par défaut (baseCache)

L'instance de mémoire cache dynamique par défaut est créée par le service de mise en mémoire cache dynamique WebSphere Application Server. Cette instance de mémoire cache dynamique de servlet est utilisée par des produits tels qu'IBM WebSphere Commerce. Contrairement aux autres instances de mémoire cache définies avec WebSphere Application Server, baseCache est propre à un seul serveur ou à une seule instance de cluster. Utilisez cette procédure pour configurer l'instance baseCache dans WebSphere Application Server afin de l'utiliser comme fournisseur de cache dynamique avec WebSphere eXtreme Scale.

Avant de commencer

- Pour que vous puissiez utiliser le fournisseur de cache dynamique, WebSphere eXtreme Scale Client doit être installé sur les déploiements de noeud WebSphere Application Server, et notamment le noeud du gestionnaire de déploiement. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server .
- Le domaine de service de catalogue WebSphere eXtreme Scale doit être configuré. Pour plus d'informations, voir Création de domaines de service de catalogue dans WebSphere Application Server.
-
- Si les serveurs de catalogue dans le domaine de service de catalogue utilisent SSL (Secure Sockets Layer) ou si vous voulez utiliser SSL pour un domaine de service de catalogue qui prend en charge SSL, vous devez activer la sécurité globale dans la console d'administration de WebSphere Application Server. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Cette procédure s'applique à la version 8.0 de la console d'administration de WebSphere Application Server. Ces informations peuvent varier en fonction la version de WebSphere Application Server que vous utilisez.

Remarque :

- WebSphere eXtreme Scale version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.
- La procédure suivante s'applique à la topologie de mémoire cache dynamique WebSphere eXtreme Scale distante. Toutes les autres topologies, notamment intégrées, partitionnées intégrées et locales, sont obsolètes dans WebSphere eXtreme Scale version 8.6.
- La procédure ci-dessous suppose que vous utilisez WebSphere Application Server version 7.0 groupe de correctifs 27, version 8.0 groupe de correctifs 6, version 8.5 groupe de correctifs 2 ou ultérieur. L'APAR PM71992 WebSphere Application Server est inclus dans ces versions.

Procédure

1. Démarrez la console d'administration WebSphere Application Server.
2. Dans le menu supérieur, cliquez sur **Serveurs > Type de serveur > Serveurs d'applications WebSphere**.
3. Dans la zone **Serveurs d'applications**, sélectionnez le **nom de votre serveur**.
4. Dans le panneau **Configuration**, cliquez sur **Services de conteneur** et sélectionnez **Service de cache dynamique**.
5. Dans la liste déroulante **Fournisseur de cache**, sélectionnez WebSphere eXtreme Scale.
6. Si vous voulez changer la taille du cache, définissez-la dans la zone de **taille de cache**. La taille de cache définit le nombre maximal d'entrées autorisées dans chaque partition dans une grille WebSphere eXtreme Scale pour l'instance de mémoire cache dynamique. La valeur par défaut est de 2000 entrées dans chaque partition.
7. Sélectionnez **Activer la réplication de cache**. Dans ce cas, les données en mémoire cache sont stockées à distance dans la grille de données et non pas localement. Lorsque vous utilisez WebSphere eXtreme Scale comme fournisseur de cache, vous devez sélectionner cette option.
8. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
9. Dans le menu supérieur, cliquez sur **Serveurs > Type de serveur > Serveurs d'applications WebSphere**.
10. Dans la zone **Serveurs d'applications**, sélectionnez le **nom de votre serveur**.
11. Dans le panneau **Configuration**, cliquez sur **Paramètres de conteneur Web** et sélectionnez **Conteneur Web**.

12. Cochez l'option d'**activation de la mise en cache de servlet**.
13. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.

Que faire ensuite

Par défaut, chaque instance de mémoire cache dynamique configurée sur WebSphere Application Server correspond à une grille de données de cache dynamique dont le nom correspond au nom JNDI de l'instance de mémoire cache. En outre, par défaut, les données de cette instance de mémoire cache sont stockées dans une mappe dynamique au sein de cette grille, et le suffixe du nom de cette mappe dynamique correspond également au nom JNDI de l'instance de mémoire cache. Par exemple, si vous configurez sur WebSphere Application Server une instance de mémoire cache portant le nom JNDI cache1, une grille de données de cache dynamique est créée sur le dispositif avec le nom cache1. Dans cette grille de données cache1, une mappe dynamique baptisée IBM_DC_PARTIONED_cache1 est créée pour stocker les données.

Dans la plupart des cas, il n'est pas nécessaire de modifier cette configuration. Toutefois, dans certaines circonstances, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant des noms JNDI différents, mappent vers différentes mappes dynamiques au sein de la même instance de grille de données. Dans d'autres cas, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant le même nom JNDI, mappent vers différentes instances de grille de données de cache dynamique ou vers différentes instances de mappe dynamique au sein de la même grille de données de cache dynamique. Par exemple, si vous avez une application qui utilise l'instance de mémoire cache dynamique par défaut (baseCache), vous pouvez vouloir utiliser le même dispositif pour vos environnements de test et de production, tout en gardant les données mises en cache dans des grilles de données distinctes ou dans des mappes dynamiques distinctes au sein de la même grille de données.

Pour contrôler cette configuration, vous pouvez définir les propriétés personnalisées suivantes pour les instances de mémoire cache :

Conseil : Ces propriétés peuvent être particulièrement utiles lorsque vous utilisez l'instance de mémoire cache dynamique par défaut (baseCache), car le nom JNDI baseCache est automatiquement attribué au cache et ne peut être modifié.

com.ibm.websphere.xs.dynacache.grid_name

Utilisez cette propriété personnalisée pour indiquer le nom de l'instance de grille de données de cache dynamique à laquelle une instance de mémoire cache dynamique correspond.

com.ibm.websphere.xs.dynacache.cache_name

Utilisez cette propriété personnalisée pour indiquer le nom à utiliser à la place du nom JNDI comme nom de la grille de données de cache dynamique et comme nom de la mappe dynamique au sein de cette grille. Si la propriété **`com.ibm.websphere.xs.dynacache.grid_name`** est également définie, cette propriété ne s'applique qu'au nom de la mappe dynamique.

Pour savoir comment définir ces propriétés, voir Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées.

Rubrique parent : Java 2.5+ Configuration des instances de cache dynamique

Java

Configuration des instances de cache dynamique d'objet ou de servlet

WebSphere Application Server permet de configurer des instances de cache dynamique d'objet ou de servlet en plus de l'instance par défaut. Pour configurer ces instances supplémentaires, procédez comme suit.

Avant de commencer

- Pour pouvoir utiliser le fournisseur de cache dynamique, WebSphere eXtreme Scale doit être installé sur les déploiements de noeud WebSphere Application Server et notamment le noeud du gestionnaire de déploiement. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server .
- Un domaine de service de catalogue WebSphere eXtreme Scale doit être configuré. Voir Création de domaines de service de catalogue dans WebSphere Application Server.
- Si les serveurs de catalogue dans le domaine de service de catalogue utilisent SSL (Secure Sockets Layer (SSL) ou que vous voulez utiliser SSL pour un domaine de service de catalogue qui le prend en charge, vous devez activer la sécurité globale dans la console d'administration WebSphere Application Server. Pour plus d'informations sur la configuration de la sécurité globale, voir Paramètres de sécurité globale.

Pourquoi et quand exécuter cette tâche

Dans cette procédure, vous pouvez créer deux types d'instance de cache : instance de cache d'objet et instance de cache de servlet. Une instance de cache d'objet est un emplacement en complément du cache dynamique partagé par défaut où les applications Java 2 Platform, Enterprise Edition (J2EE) peuvent stocker, distribuer et partager des objets. Après avoir configuré des instances de cache d'objet, vous pouvez utiliser l'interface DistributedMap ou DistributedObjectCache dans le package com.ibm.websphere.cache pour accéder par programme aux instances de cache. Voir API (Application Programming Interface) supplémentaires pour plus d'informations sur les interfaces DistributedMap et DistributedObjectCache . Les instances de cache de servlet sont des emplacements qui, en plus du cache dynamique par défaut, permettent au cache dynamique de stocker, distribuer et partager le résultat et les effets secondaires d'un servlet appelé. En configurant une instance de cache de servlet, vous conférez aux applications une souplesse plus grande et les ressources en mémoire cache sont mieux optimisées. Le nom JNDI (Java Naming and Directory Interface) défini pour l'instance de cache dans la console d'administration est associé à l'élément d'instance de cache dans le fichier de configuration `cachespec.xml`. Tous les éléments <cache-entry> spécifiés dans un élément <cache-instance> sont créés dans cette instance de cache spécifique. Tous les éléments <cache-entry> spécifiés en dehors d'un élément <cache-instance> sont stockés dans l'instance de cache dynamique par défaut. Voir Instances de cache pour plus d'informations sur les instances de cache de type objet et servlet.

Cette procédure s'applique à la version 8.0 de la console d'administration WebSphere Application Server. Ces informations peuvent varier en fonction la version WebSphere Application Server que vous utilisez.

Remarque :

- WebSphere eXtreme Scale version 8.6 n'est pas pris en charge sur les versions de WebSphere Application Server antérieures à la version 7.0.

Procédure

- Pour configurer un cache d'objet ou de servlet avec la console d'administration WebSphere Application Server, procédez comme suit :

  1. Démarrez la console d'administration WebSphere Application Server.
  2. Dans le menu supérieur, cliquez sur **Ressources > Instances de cache > Instances de cache d'objet**.
  3. Dans la zone des **instances de cache d'objet**, sélectionnez le type d'instance de cache à créer. Il peut s'agir d'une instance de cache d'objet ou de servlet.
  4. Définissez la portée de l'instance de cache. Spécifiez une portée de cellule pour que l'instance de cache soit disponible pour tous les serveurs de la cellule. Une portée de noeud rend l'instance de cache disponible pour tous les serveurs d'un noeud. Une portée de serveur rend l'instance de cache disponible pour le serveur sélectionné uniquement. Si nécessaire, vous pouvez combiner les portées.
  5. Cliquez sur **Appliquer** et enregistrez la portée.
  6. Cliquez sur **Nouveau** et définissez une instance de cache d'objet.
  7. Dans la liste déroulante **Fournisseur de cache**, sélectionnez WebSphere eXtreme Scale.

     Remarque : Si WebSphere eXtreme Scale n'apparaît pas comme fournisseur de cache

dynamique, cela implique que le profil WebSphere Application Server n'a pas été étendu pour WebSphere eXtreme Scale.

8. Spécifiez le nom JNDI de l'instance de cache dynamique. Pour les objets cache, ce nom sera utilisé lors de la consultation du cache. Pour les caches de servlet, il s'agit du nom d'attribut défini dans l'élément <cache-instance> dans le fichier `cachespec.xml`.
9. Spécifiez le nom JNDI de l'instance de cache dynamique.
10. Si vous voulez changer la taille du cache, redéfinissez-la dans la zone de **taille de cache**. La taille de cache définit le nombre maximal d'entrées autorisées dans chaque partition dans une grille WebSphere eXtreme Scale pour l'instance de cache dynamique. La valeur par défaut est 2000 entrées dans chaque partition.
11. Cochez la case d'**activation de la réplication de cache**. Dans ce cas, les données en cache sont stockées à distance dans la grille et non pas localement. Vous devez cocher cette case lorsque vous utilisez WebSphere eXtreme Scale comme fournisseur de cache.
12. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.

- Pour configurer un cache d'objet ou de servlet en utilisant le fichier `cacheinstances.properties`, procédez comme suit :

  1. Créez un fichier `cacheinstances.properties`. Voir Fichier de propriétés des instances de cache pour le contenu nécessaire.
  2. Placez le fichier `cacheinstances.properties` dans le serveur d'applications ou le chemin d'accès aux applications. Par exemple, vous pouvez utiliser votre fichier WAR (Web application archive) d'application, le répertoire `WEB-INF\classes` ou créer un répertoire `server_root\classes` pour l'y placer.

Que faire ensuite

Par défaut, chaque instance de mémoire cache dynamique configurée sur WebSphere Application Server correspond à une grille de données de cache dynamique dont le nom correspond au nom JNDI de l'instance de mémoire cache. En outre, par défaut, les données de cette instance de mémoire cache sont stockées dans une mappe dynamique au sein de cette grille, et le suffixe du nom de cette mappe dynamique correspond également au nom JNDI de l'instance de mémoire cache. Par exemple, si vous configurez sur WebSphere Application Server une instance de mémoire cache portant le nom JNDI cache1, une grille de données de cache dynamique est créée sur le dispositif avec le nom cache1. Dans cette grille de données cache1, une mappe dynamique baptisée IBM_DC_PARTIONED_cache1 est créée pour stocker les données.

Dans la plupart des cas, il n'est pas nécessaire de modifier cette configuration. Toutefois, dans certaines circonstances, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant des noms JNDI différents, mappent vers différentes mappes dynamiques au sein de la même instance de grille de données. Dans d'autres cas, vous pouvez avoir besoin que plusieurs instances de mémoire cache, portant le même nom JNDI, mappent vers différentes instances de grille de données de cache dynamique ou vers différentes instances de mappe dynamique au sein de la même grille de données de cache dynamique. Par exemple, si vous avez une application qui utilise l'instance de mémoire cache dynamique par défaut (baseCache), vous pouvez vouloir utiliser le même dispositif pour vos environnements de test et de production, tout en gardant les données mises en cache dans des grilles de données distinctes ou dans des mappes dynamiques distinctes au sein de la même grille de données.

Pour contrôler cette configuration, vous pouvez définir les propriétés personnalisées suivantes pour les instances de mémoire cache :

Conseil : Ces propriétés peuvent être particulièrement utiles lorsque vous utilisez l'instance de mémoire cache dynamique par défaut (baseCache), car le nom JNDI baseCache est automatiquement attribué au cache et ne peut être modifié.

com.ibm.websphere.xs.dynacache.grid_name

Utilisez cette propriété personnalisée pour indiquer le nom de l'instance de grille de données de cache dynamique à laquelle une instance de mémoire cache dynamique correspond.

com.ibm.websphere.xs.dynacache.cache_name

Utilisez cette propriété personnalisée pour indiquer le nom à utiliser à la place du nom JNDI comme nom de la grille de données de cache dynamique et comme nom de la mappe dynamique au sein de cette grille. Si la propriété **`com.ibm.websphere.xs.dynacache.grid_name`** est également définie, cette propriété ne s'applique qu'au nom de la mappe dynamique.

Pour savoir comment définir ces propriétés, voir Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées.

Rubrique parent : Java 2.5+ Configuration des instances de cache dynamique

Java

Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées

WebSphere Application Server permet de définir des propriétés personnalisées dans une instance de cache dynamique.

Avant de commencer

Vous devez avoir configuré une instance de cache par défaut ou disposer d'un objet supplémentaire ou d'une instance de cache de type servlet. Voir Configuration de l'instance de mémoire cache dynamique par défaut (baseCache) or Configuration des instances de cache dynamique d'objet ou de servlet.

Pourquoi et quand exécuter cette tâche

Vous pouvez définir des propriétés personnalisées spécifiques d'une instance de cache dynamique en procédant comme suit :

- Utilisez la console d'administration WebSphere Application Server avec l'APAR PM71992 si vous voulez définir des propriétés spécifiques d'une instance de cache dynamique donnée. Si vous ne disposez pas de ce correctif, contactez WebSphere Application Server sur la page du support http://www.ibm.com/software/webservers/appserv/was/support.
- Si vous avez créé un fichier `cacheinstances.properties`, vous pouvez définir des propriétés personnalisées dans ce fichier. Cette méthode ne peut pas être utilisée pour définir des propriétés personnalisées pour une instance de cache dynamique (baseCache) par défaut.
- Utilisez la console d'administration WebSphere Application Server pour changer les valeurs des propriétés personnalisées JVM (Java virtual machine).

  Remarque : Les propriétés JVM peuvent affecter toutes les instances de cache dans une machine JVM donnée.

  Remarque : La portée des propriétés JVM peut s'appliquer à toutes les instances de cache dans une machine JVM WebSphere Application Server JVM. Par conséquent, il est préférable d'utiliser des propriétés personnalisées de cache dans la console d'administration WebSphere Application Server (avec l'APAR PM71992 pour une instance de cache par défaut), ou un fichier `cacheinstances.properties` dans la plupart des cas.

Procédure

- Pour définir une propriété personnalisée dans une instance de cache dans la console d'administration WebSphere Application Server, procédez comme suit :
  1. Démarrez la console d'administration WebSphere Application Server.

     Remarque : Ces étapes ne peuvent pas être exécutées pour une instance (baseCache) par défaut tant que vous n'avez pas appliqué l'APAR R PM71992 WebSphere Application Server. Ce correctif est disponible dans les versions WebSphere Application Server 7.0.0.27, 8.0.0.6, 8.5.0.2 et les versions suivantes. Si vous ne disposez pas de correctif, consultez la page WebSphere Application Server Support, http://www.ibm.com/software/webservers/appserv/was/support.
  2. Accédez à l'instance de cache désirée que vous avez configurée.
  3. Dans le panneau d'instance de cache, cliquez sur **Propriétés supplémentaires > Propriétés personnalisées**.
  4. Sélectionnez **Nouveau** et définissez le nom et la valeur de la propriété personnalisée.
  5. Cliquez sur **Appliquer** ou **OK** et enregistrez la configuration.
  6. Redémarrez le gestionnaire de déploiement et tous les processus serveur d'applications.
- Pour définir une propriété personnalisée pour une instance de cache en utilisant le fichier `cacheinstances.properties`, procédez comme suit :

  Remarque : Ces étapes ne peuvent pas être exécutées pour une instance (baseCache) par défaut.

  1. Ajoutez la propriété personnalisée à un fichier `cacheinstances.properties`. Si vous devez créer ce fichier, voir Fichier de propriétés des instances de cache pour connaître le contenu requis.
  2. Placez le fichier `cacheinstances.properties` dans le serveur d'applications ou le chemin d'accès aux applications. Par exemple, vous pouvez utiliser votre fichier WAR (Web application archive) d'application, le répertoire `WEB-INF\classes` ou créer un répertoire `server_root\classes` pour l'y placer.
- Utilisez la console d'administration WebSphere Application Server pour changer les valeurs des

propriétés personnalisées JVM (Java virtual machine). Voir Propriétés personnalisées JVM (java virtual machine) pour plus d'informations.

Java 2.5+ **Fichier de propriétés des instances de cache**

Pour configurer un cache d'objet ou de servlet en utilisant le fichier `cacheinstances.properties`.

Java 2.5+ **Propriétés personnalisées de cache dynamique**

Reportez-vous à ce tableau pour définir les propriétés personnalisées d'une instance de cache dynamique par défaut ou d'une instance de cache de servlet ou d'objet.

Rubrique parent : Java 2.5+ Configuration des instances de cache dynamique

Fichier de propriétés des instances de cache

Pour configurer un cache d'objet ou de servlet en utilisant le fichier `cacheinstances.properties`.

Tableau 1. Propriétés des instances de cache

Nom de la propriété : – x est le numéro de l'instance	Obligatoire	Portée	Valeur possible	Description
cache.instance.x	Oui	Par instance de cache	n'importe quelle chaîne (pas de définition par défaut)	Spécifie le nom de l'instance du cache ou le nom JNDI.
cache.instance.x.cacheSize	Non	Par instance de cache	> 0 (par défaut =2000)	Indique le nombre maximal d'entrées autorisées dans une partition dans la grille WebSphere eXtreme Scale. Multipliez cette valeur par le nombre de partitions pour déterminer la capacité du cache dans la grille WebSphere eXtreme Scale.
cache.instance.x.createCacheAtServerStartup	Non	Par instance de cache	True ou false (par défaut=false)	Indique si l'instance de cache configurée est créée lors du démarrage du serveur.
cache.instance.x.enableServletSupport	Non	Par instance de cache	True ou false (par défaut=false)	Spécifie si l'instance de cache correspond au cache des servlets ou au cache d'objets.
cache.instance.x.enableCacheReplication	Oui (uniquement jusqu'à l'APAR)	Par instance de cache	True ou false (par défaut=false)	Indique que le cache est éloigné du serveur d'applications. Cette propriété doit avoir la valeur True dans la topologie distante WebSphere eXtreme Scale.
cache.instance.x.cacheProviderName	Oui	Par instance de cache	com.ibm.ws.objectgrid.dynacache.CacheProviderImpl	Indique le fournisseur de cache dynamique. Le fournisseur WebSphere Application Server est utilisé par défaut si WebSphere eXtreme Scale n'est pas défini.
cache.instance.x.ignoreValueInInvalidationEvent	Non	Par instance de cache	True ou false (par défaut=false)	Spécifie si la valeur de cache de l'événement d'invalidation est

		cache		à invalidation est ignorée. Si la propriété est définie comme true, la valeur en cache de l'événement est défini comme NULL lorsque le code est retourné à l'appelant.
cache.instance.x.<custom property>	Dépend de la propriété à ajouter.	Par instance de cache	Dépend de la propriété à ajouter.	Vous pouvez ajouter n'importe quelle propriété à ce fichier.

Rubrique parent : Java 2.5+ Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées

Propriétés personnalisées de cache dynamique

Reportez-vous à ce tableau pour définir les propriétés personnalisées d'une instance de cache dynamique par défaut ou d'une instance de cache de servlet ou d'objet.

Tableau 1. Propriétés personnalisées de cache dynamique

Nom de la propriété	Obligatoire	Portée	Valeur possible	Description
com.ibm.websphere.xs.dynacache.topology	Oui (après l'application de l'APAR PM71992 de WebSphere Application Server, cette propriété n'est plus nécessaire.)	Par instance de cache	`remote`	Indique la topologie de l'instance de cache. La topologie `remote` est la seule topologie utilisable avec WebSphere DataPower XC10 Appliance.
com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent	Non	Par instance de cache	`true` ou `false` Valeur par défaut : `true`	Spécifie si la valeur en cache de l'événement d'invalidation est ignorée. Si la propriété est définie comme true, la valeur en cache de l'événement est défini comme NULL lorsque le code est retourné à l'appelant.
com.ibm.websphere.xs.dynacache.ignore_value_in_change_event	Non	Par instance de cache	`true` ou `false` Valeur par défaut : `true`	Spécifie si la valeur en cache de l'événement de changement est ignorée. Si la valeur est true, la valeur de cache de l'événement de changement est NULL lorsque le code est retourné à l'appelant.
com.ibm.websphere.xs.dynacache.cs_override	Non	Par instance de cache	Noeud final de service de catalogue Exemple : `9.5.12.345:2819`	Indique le noeud final de service de catalogue de la grille de données à associer à cette instance de cache. Cette zone est obligatoire si elle n'est pas spécifiée dans la console d'administration

				d'administration WebSphere Application Server.
com.ibm.websphere.xs.dynacache.grid_name	Non	Par instance de cache	N'importe quelle chaîne Valeur par défaut : `nom JNDI de l'instance de cache`	Indique le nom de la grille de données que vous avez créée.
com.ibm.websphere.xs.dynacache.map_template_name	Non	Par instance de cache	Vous pouvez utiliser l'un des modèles de mappe suivants : • `IBM_DC_PARTITIONED_.*` (valeur par défaut) • `IBM_DC_NCI_PARTITIONED_.*` (Indique que ce cache utilise un cache local.)	Indique le nom du préfixe de mappe de modèle.
com.ibm.websphere.xs.dynacache.cache_name	Non	Par instance de cache	N'importe quelle chaîne Valeur par défaut : `valeur de cache.instance.x`	Définit le nom du suffixe unique qui est utilisé comme nom de la mappe de modèle. Par exemple, `IBM_DC_PARTITIONED.<nom_cache>`
com.ibm.websphere.xs.dynacache.near_cache_size	Non	Par instance de cache	Valeur supérieur à zéro. Valeur par défaut : `valeur spécifiée dans cache.instance.x.cacheSize`	Indique le nombre maximal d'entrées autorisées dans une instance de cache local. Par défaut, cette valeur est égale au nombre maximal d'entrées autorisées dans une partition dans l'ObjectGrid distant de cette instance de cache.
com.ibm.websphere.xs.dynacache.request_retry_timeout_override	Non	Par instance de cache	Valeur entière représentant des millisecondes : • `-1` : Infini. Les requêtes n'expirent jamais. • `0` : Les requêtes expirent	Définit le temps (en millisecondes) pendant lequel une requête peut s'exécuter avant qu'un dépassement de délai ne se produise. Cette propriété remplace la propriété **`requestRetryTimeout`** si elle est définie dans le fichier de propriétés

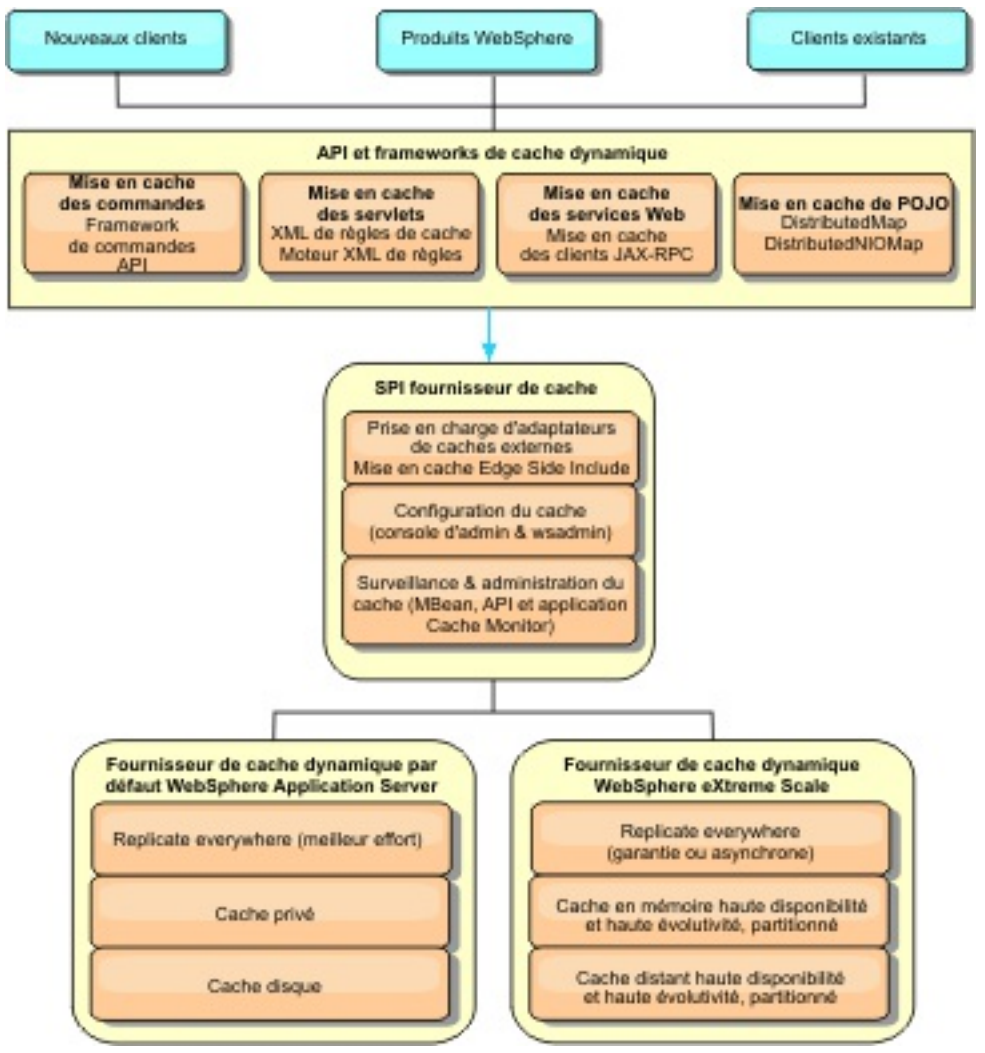
				immédiatement. • >0 : Délai (en millisecondes) pendant lequel les requêtes peuvent s'exécuter avant d'expirer. • 2000 : Valeur par défaut adoptée si vous n'utilisez pas cette propriété personnalisée ou le fichier de propriétés du client pour définir cette valeur.	du client. Pour plus d'informations, voir Fichier de propriétés du client. Le délai d'attente par défaut pour les nouvelles tentatives d'exécution des requêtes pour les instances de cache dynamique est de 2000 millisecondes s'il n'est pas défini à l'aide de cette propriété personnalisée ou dans le fichier des propriétés du client.

Rubrique parent : Java 2.5+ Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées

Présentation du fournisseur de cache dynamique

Java WebSphere Application Server fournit un service de cache dynamique disponible pour déployer des applications Java™ EE. Ce service permet de mettre des données en cache, telles que la sortie d'un servlet, une page JSP ou des commandes, ainsi que les données d'objet définies par programme dans une application d'entreprise en utilisant des API DistributedMap. En configurant cette fonction, vous pouvez activer les applications qui utilisent le service de cache dynamique pour qu'elles utilisent les fonctions et les fonctionnalités de performance de WebSphere DataPower XC10 Appliance.

Initialement, le seul fournisseur de service pour le service de cache dynamique était le moteur de cache dynamique par défaut intégré dans WebSphere Application Server. Vous pouvez également définir WebSphere DataPower XC10 Appliance comme fournisseur de cache pour une instance de cache. En configurant cette fonction, vous pouvez activer des applications qui utilisent le service de mise en mémoire cache dynamique, pour utiliser les fonctions et les fonctions de performances de .



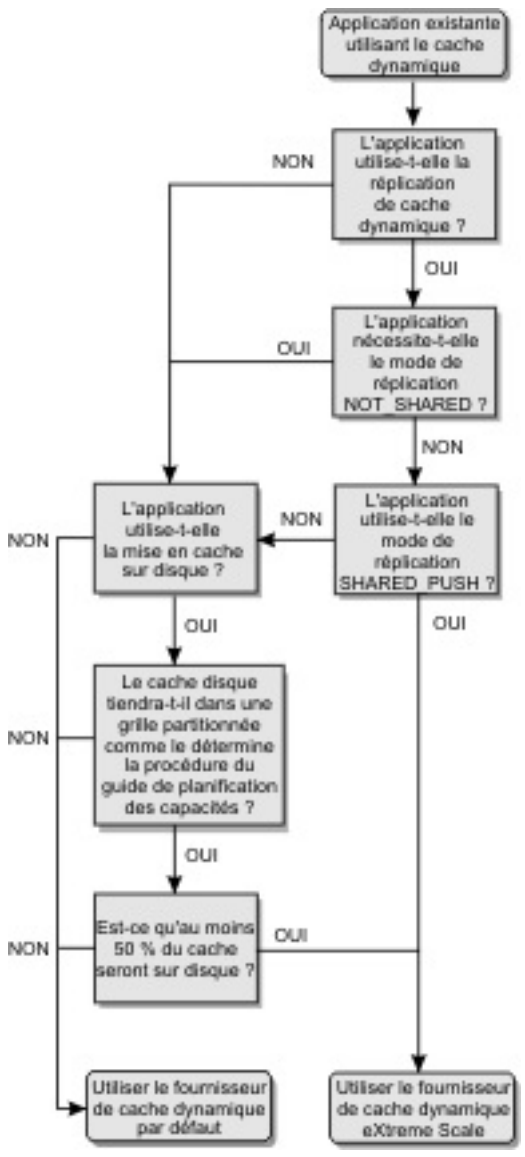
Vous pouvez installer et configurer le fournisseur de cache dynamique comme décrit dans Configuration de l'instance de mémoire cache dynamique par défaut (baseCache).

Choix du mode d'utilisation de WebSphere DataPower XC10 Appliance

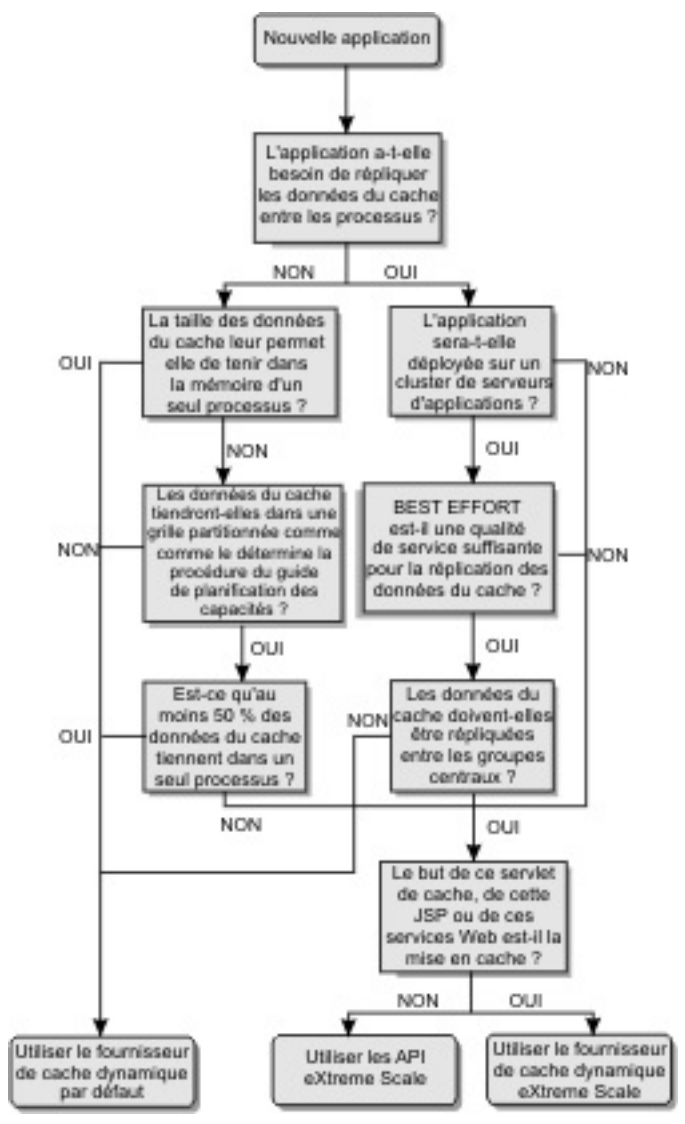
Les fonctions disponibles dans WebSphere DataPower XC10 Appliance améliorent sensiblement les fonctionnalités distribuées du service de cache dynamique par rapport au fournisseur de cache dynamique par défaut et les service de réplication de données. Avec eXtreme Scale, vous pouvez créer des mémoires cache véritablement réparties entre plusieurs serveurs et non simplement répliquées et synchronisées d'un serveur à l'autre. Par ailleurs, les mémoires cache eXtreme Scale sont transactionnelles et hautement disponibles : chaque serveur voit ainsi le même contenu pour le service de cache dynamique. WebSphere DataPower XC10 Appliance offre une qualité de service supérieure pour la réplication de cache fournie via DRS.

Tous ces avantages ne signifient cependant pas que le fournisseur de cache dynamique eXtreme Scale constitue la meilleure solution pour toutes les applications. Pour identifier la technologie la mieux adaptée à votre application, utilisez l'arbre de décision et la matrice de comparaison des fonctions.

Arbre de décision permettant de faire migrer des applications existantes de cache dynamique



Arbre de décision permettant de choisir un fournisseur de cache pour les nouvelles applications.



Comparaison des fonctionnalités

Tableau 1. Comparaison des fonctionnalités

Fonctionnalités de cache	Fournisseur par défaut	Fournisseur eXtreme Scale	API eXtreme Scale
Mise en cache local en mémoire	Oui	Via le cache local	Via le cache local
Mise en cache réparti	Via DRS	Oui	Oui

Evolutivité linéaire	Non	Oui	Oui
Réplication fiable (synchrone)	Non	Oui	Oui
Dépassement de capacité des disques	Oui	N/A	N/A
Expulsion	LRU/TTL/en fonction des segments	LRU/TTL (par partition)	LRU/TTL (par partition)
Invalidation	Oui	Oui	Oui
Relations	Relations d'ID de modèle/dépendance	Oui	Non (d'autres relations sont possibles)
Recherches non-clés	Non	Non	Via l'interrogation et l'index
Intégration dorsale	Non	Non	Via les chargeurs
Transactionnel	Non	Oui	Oui
Stockage à base de clés	Oui	Oui	Oui
Evénements et programmes d'écoute	Oui	Non	Oui
Intégration à WebSphere Application Server	Une seule cellule	Plusieurs cellules	Indépendant des cellules
Prise en charge de Java Standard Edition	Non	Oui	Oui
Surveillance et statistiques	Oui	Oui	Oui
Sécurité	Oui	Oui	Oui

Remarque : Le cache eXtreme Scale réparti peut uniquement stocker des entrées dont la clé et la valeur implémentent toutes les deux l'interface java.io.Serializable.

Moteur de cache dynamique et différences fonctionnelles avec eXtreme Scale

Les utilisateurs ne constatent aucune différence, sinon que les caches WebSphere DataPower XC10 Appliance ne supportent pas le déchargement sur disque ni les statistiques ou opérations en rapport avec la taille du cache en mémoire.

Il n'existe pas de différence notable dans les résultats retournés par la plupart des appels d'API de cache, que vous utilisiez le fournisseur de cache dynamique par défaut ou le fournisseur de cache eXtreme Scale.

Pour certaines opérations, il est impossible d'émuler le comportement du moteur de cache dynamique à l'aide d'eXtreme Scale.

Statistiques du cache dynamique

Appels des beans gérés

Le fournisseur de cache dynamique WebSphere eXtreme Scale ne prend pas en charge la mise en cache sur un disque. Les appels de beans gérés relatifs à une mise en cache sur un disque ne fonctionnent pas.

Mappage des règles de réplication du cache dynamique

La topologie distante du fournisseur de cache dynamique eXtreme Scale prend en charge une règle de réplication qui est très similaire aux règles SHARED_PULL et SHARED_PUSH_PULL (dans la terminologie utilisée par le fournisseur de cache dynamique WebSphere Application Server par défaut). Dans un cache dynamique eXtreme Scale, l'état distribué du cache est cohérent entre tous les serveurs.

Invalidation de l'index global

Vous pouvez utiliser un index global pour améliorer l'efficacité de l'invalidation dans les grands environnements partitionnés comportant, par exemple, plus de 40 partitions. Sans l'index global, le modèle de cache dynamique et le traitement de l'invalidation de dépendance doivent envoyer des demandes d'agent distant à toutes les partitions, ce qui affecte les performances. Lorsque vous configurez un index global, des agents d'invalidation sont envoyés uniquement aux partitions concernées qui contiennent des entrées de cache associées à l'ID de modèle ou de dépendance. Les possibilités d'amélioration des performances seront plus importantes dans les environnements comportant un grand nombre de partitions configurées. Vous pouvez configurer un index global en utilisant les index d'ID de dépendance et d'index qui sont disponibles dans les exemples de fichiers XML descripteurs objectGrid de cache dynamique.

Cache local

Vous pouvez configurer une instance de cache dynamique pour créer et gérer un cache local qui résidera dans la machine JVM du serveur d'applications. Ce cache contient un sous-ensemble des entrées contenues dans l'instance de cache dynamique distant. Pour plus d'informations, voir Configuration d'un cache local pour la mémoire cache dynamique. Il existe des propriétés personnalisées qui permettent d'optimiser le cache local. Pour plus d'informations, voir Propriétés personnalisées de cache dynamique.

2.5+

Réplication multimaître

Vous pouvez choisir d'activer la réplication multimaître dans le fournisseur de cache dynamique WebSphere DataPower XC10 Appliance. Pour plus d'informations, voir Création de grilles de données en cache dynamique.

Informations supplémentaires

- Redbook relatif au cache dynamique
- Documentation relative au cache dynamique
    - WebSphere Application Server 7.0
- Documentation relative à DRS
    - WebSphere Application Server 7.0

Rubrique parent : Java Création de grilles de données en cache dynamique

Java

Création de domaines de service de catalogue dans WebSphere Application Server

A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données de cache dynamique.

Avant de commencer

- La création de domaine de service de catalogue n'est requis que pour les grilles de données à caches dynamiques. Si vous utilisez des grilles de données simples ou des grilles de données de session, vous n'avez pas besoin de configurer de domaine de service de catalogue. WebSphere eXtreme Scale Client doit être installé sur WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

En créant un domaine de service de catalogue, vous définissez une collection de serveurs de catalogue à haute disponibilité. En configurant un domaine de service de catalogue, vous établissez des connexions aux serveurs de catalogue exécutés sur WebSphere DataPower XC10 Appliance. Ce domaine de service de catalogue représente le groupe des serveurs de catalogue qui s'exécutent sur les dispositifs de votre collectivité.

Procédure

1. Créez le domaine de service de catalogue.
    a. Dans la console d'administration de WebSphere Application Server, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaines de services de catalogue** > **Nouveau**.
    b. Définissez un nom, une valeur par défaut et des justificatifs d'identification pour l'authentification JMX de votre domaine.
    c. Ajoutez des points de contact de serveurs de catalogue.

       Spécifiez un groupe de serveurs distants qui sont les serveurs de catalogue qui s'exécutent sur les dispositifs de la collectivité. Pour visualiser les serveurs de catalogue qui s'exécutent dans la collectivité, cliquez sur **Collectivité** > **Membres** > ***nom_membre***. La zone **Serveurs de catalogue** affiche la liste des serveurs de catalogue en cours d'exécution dans la collectivité. Vous devez spécifier les points de contact avec leurs adresses IP ou leurs noms d'hôtes qualifiés complets. Utilisez les valeurs de port suivantes pour les dispositifs :
        - **Port de client** : non requis pour les connexions aux serveurs de catalogue exécutés sur le dispositif.
        - **Port d'écoute** : `2809`
2. Testez la connexion aux serveurs de catalogue dans le domaine de service de catalogue.
    a. Dans la console d'administration de WebSphere Application Server, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaines de services de catalogue**.
    b. Sélectionnez le domaine que vous voulez tester et cliquez sur **Tester la connexion**. Lorsque vous cliquez sur ce bouton, tous les points de contact des domaines de service de catalogue définis sont interrogés l'un après l'autre (s'il existe des points de contact) et la procédure retourne un message indiquant que la connexion au domaine a réussi.

Java **Tâches d'administration des domaines de service de catalogue**

Les langages de script Jacl ou Jython permettent de gérer les domaines de service de catalogue présents dans votre configuration WebSphere Application Server. A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données à caches dynamiques.

Rubrique parent : Java Création de grilles de données en cache dynamique

Java

Tâches d'administration des domaines de service de catalogue

Les langages de script Jacl ou Jython permettent de gérer les domaines de service de catalogue présents dans votre configuration WebSphere Application Server. A l'aide de WebSphere DataPower XC10 Appliance, vous pouvez définir des domaines de service de catalogue pour établir des connexions avec les serveurs de catalogue exécutés sur le dispositif. La création de cette configuration n'est requise que pour les grilles de données à caches dynamiques.

Conditions requises

WebSphere eXtreme Scale Client doit être installé dans votre environnement WebSphere Application Server.

Afficher la liste de toutes les tâches d'administration

Pour obtenir la liste de toutes les tâches d'administration associées aux domaines de service de catalogue, exécutez la commande suivante avec **wsadmin**:

```
wsadmin>$AdminTask help XSDomainManagement
```

Commandes

Les tâches d'administration de domaines de service de catalogue comprennent les commandes suivantes :

- createXSDomain
- deleteXSDomain
- getDefaultXSDomain
- listXSDomains
- modifyXSDomain
- getTransport
- testXSDomainConnection
- testXSServerConnection

createXSDomain

La commande **createXSDomain** enregistre un nouveau domaine de service de catalogue.

Tableau 1. Arguments de la commande createXSDomain

Argument	Description
-name (requis)	Spécifie le nom du domaine de service de catalogue à créer.
-default	Indique si le domaine de service de catalogue est le domaine par défaut de la cellule. La valeur par défaut est `true`. (booléen : a soit la valeur `true`, soit la valeur `false`).
-properties	Spécifie les propriétés personnalisées du domaine de service de catalogue.
-enableXIO	Indique si IBM eXtreme IO (XIO) ou ORB (Object Request Broker) est utilisé pour la communication de transport dans ce domaine de service de catalogue. **true** Indique que XIO est utilisé. **false** Indique qu'ORB est utilisé. Si vous n'indiquez pas de valeur, la valeur par défaut est `true` (XIO activé). Si le domaine de service de catalogue contient des serveurs distants, le paramètre **-enableXIO** ne configure pas XIO ni ORB sur les serveurs distants. Pour configurer le transport sur les serveurs distants, définissez le type de transport lorsque vous les démarrez.

Tableau 2. Arguments de la procédure defineDomainServers

Argument	Description
name_o	Spécifie le nom du point de contact du service de catalogue. • **Pour les serveurs d'applications existants** : le nom du noeud final doit avoir le format *cell name**node name**server name*

f_endpoint	• **Pour les serveurs distantes** : définit le nom d'hôte du serveur distant. Vous pouvez utiliser le même nom pour plusieurs noeuds finaux, mais les valeurs de port client doivent être uniques pour chaque noeud final.
custom_properties	Spécifie les propriétés personnalisées du point de contact du domaine de service de catalogue. Si vous ne disposez pas de propriétés personnalisées, utilisez des guillemets doubles ("") pour cet argument.
endpoint_ports	Spécifie les numéros de port du point de contact du domaine de service de catalogue. Les ports doivent être définis dans l'ordre suivant : *<client_port>,<listener_port>* **Port du client** Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent uniquement dans des processus WebSphere Application Server et elle peut correspondre à n'importe quel port inutilisé autre part. **Port d'écoute** Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue. **Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance** : utilisez la valeur 2809 pour les noeuds finaux distants d'appliance.

Valeur retournée :

Exemples de mode de traitement par lots

Le mode de traitement par lots impose de formater correctement l'entrée de commande. Utilisez le mode interactif pour que les valeurs que vous entrez soient correctement traitées. Lorsque vous utilisez le mode de traitement par lots, vous devez définir les arguments d'étape **-defineDomainServers** en utilisant un tableau de propriétés spécifiques. Ce tableau a le format *name_of_endpoint custom_properties endpoint_ports*. La valeur *endpoint_ports* est la liste des ports qui doivent être définis dans l'ordre suivant : *<client_port>,<listener_port>*.

- Créez un domaine de service de catalogue de noeuds finaux distants en utilisant Jacl :

```
$AdminTask createXSDomain {-name TestDomain -default true -defineDomainServers
{{xhost1.ibm.com "" ,2809}} }}
```

- Créez un domaine de service de catalogue de noeuds finaux distants en utilisant la chaîne Jython :

```
AdminTask.createXSDomain('[-name TestDomain -default true
-defineDomainServers [[xhost1.ibm.com "" ,2809]
[xhost2.ibm.com "" ,2809]]  ]')
```

- réez un domaine de service de catalogue de noeuds finaux de serveur d'applications existants en utilisant Jacl :

```
$AdminTask createXSDomain {-name TestDomain -default true -defineDomainServers
 {{cellName/nodeName/serverName "" 1109}}}
```

Exemples de mode interactif

- Jacl :

```
$AdminTask createXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.createXSDomain ('[-interactive]')
```

deleteXSDomain

La commande **deleteXSDomain** supprime un domaine de service de catalogue.

Paramètres requis :

-name

Spécifie le nom du domaine de service de catalogue à supprimer.

Valeur retournée :

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask deleteXSDomain {-name TestDomain }
```

- A l'aide de la chaîne Jython :

```
AdminTask.deleteXSDomain('[-name TestDomain ]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask deleteXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.deleteXSDomain ('[-interactive]')
```

getDefaultXSDomain

La commande **getDefaultXSDomain** retourne le domaine de service de catalogue par défaut de la cellule.

Paramètres requis : aucun.

Valeur retournée : le nom du domaine de service de catalogue par défaut.

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask getDefaultXSDomain
```

- A l'aide de la chaîne Jython :

```
AdminTask.getDefaultXSDomain
```

Exemples de mode interactif

- Jacl :

```
$AdminTask getDefaultXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.getDefaultXSDomain ('[-interactive]')
```

listXSDomains

La commande **listXSDomains** retourne la liste des domaines de service de catalogue existants.

Paramètres requis : aucun.

Valeur retournée : la liste de tous les domaines de service de catalogue présents dans la cellule.

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask listXSDomains
```

- A l'aide de la chaîne Jython :

```
AdminTask.listXSDomains
```

Exemples de mode interactif

- Jacl :

```
$AdminTask listXSDomains {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.listXSDomains ('[-interactive]')
```

modifyXSDomain

La commande **modifyXSDomain** modifie un domaine de service de catalogue existant.

Le mode de traitement par lots impose de formater correctement l'entrée de commande. Utilisez le mode interactif pour que les valeurs que vous entrez soient correctement traitées. Lorsque vous utilisez le mode de traitement par lots, vous devez définir les arguments d'étape **-modifyEndpoints**, **-addEndpoints** et **-removeEndpoints** en utilisant un tableau de propriétés spécifiques. Ce tableau a le format *name_of_endpoint host_name custom_properties endpoint_ports*. La valeur *endpoint_ports* est la liste des ports qui doivent être définis dans l'ordre suivant : *<client_port>*,*<listener_port>*.

Tableau 3. Arguments de la commande modifyXSDomain

Argument	Description
-name (requis)	Spécifie le nom du domaine de service de catalogue que vous souhaitez éditer.
-default	Avec la valeur `true`, spécifie que le domaine de service de catalogue est le domaine par défaut de la cellule (booléen).
-properties	Spécifie les propriétés personnalisées du domaine de service de catalogue.
-enableXIO	Indique si IBM eXtreme IO (XIO) ou ORB (Object Request Broker) est utilisé pour la communication de transport dans ce domaine de service de catalogue. **true** Indique que XIO est utilisé. **false** Indique qu'ORB est utilisé. Si vous n'indiquez pas de valeur, la valeur par défaut est `true` (XIO activé). Si le domaine de service de catalogue contient des serveurs distants, vous ne pouvez pas configurer XIO sur les serveurs distants.

Tableau 4. Arguments de la procédure modifyEndpoints

Argument	Description
name_of_endpoint	Spécifie le nom du point de contact du service de catalogue. • **Pour les serveurs d'applications existants** : le nom du noeud final doit avoir le format *cell_name**node_name**server_name* • **Pour les serveurs distantes** : définit le nom d'hôte du serveur distant. Vous pouvez utiliser le même nom pour plusieurs noeuds finaux, mais les valeurs de port client doivent être uniques pour chaque noeud final. Cette valeur doit être un nom qualifié complet de domaine si vous configurez un dispositif.
endpoint_ports	Spécifie les numéros de port du point de contact du domaine de service de catalogue. Les noeuds finaux doivent être définis dans l'ordre suivant : *<client_port>*, *<listener_port>* **Port du client**

ts	Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent uniquement dans des processus WebSphere Application Server et elle peut correspondre à n'importe quel port inutilisé autre part. **Port d'écoute** Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue. **Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance** : utilisez la valeur 2809 pour les noeuds finaux distants d'appliance.

Tableau 5. Arguments de la procédure addEndpoints

Argument	Description
name_of_endpoint	Spécifie le nom du point de contact du service de catalogue. • **Pour les serveurs d'applications existants** : le nom du noeud final doit avoir le format *cell_name**node_name**server_name* • **Pour les serveurs distantes** : définit le nom d'hôte du serveur distant. Vous pouvez utiliser le même nom pour plusieurs noeuds finaux, mais les valeurs de port client doivent être uniques pour chaque noeud final. Cette valeur doit être un nom qualifié complet de domaine si vous configurez un dispositif.
custom_properties	Spécifie les propriétés personnalisées du point de contact du domaine de service de catalogue. Si vous ne disposez pas de propriétés personnalisées, utilisez des guillemets doubles ("") pour cet argument.
endpoint_ports	Spécifie les numéros de port du point de contact du domaine de service de catalogue. Les noeuds finaux doivent être définis dans l'ordre suivant : <*client_port*>, <*listener_port*> **Port du client** Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent uniquement dans des processus WebSphere Application Server et elle peut correspondre à n'importe quel port inutilisé autre part. **Port d'écoute** Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue. **Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance** : utilisez la valeur 2809 pour les noeuds finaux distants d'appliance.

Tableau 6. Arguments de la procédure removeEndpoints

Argument	Description
name_of_endpoint	Spécifie le nom du point de contact de domaine de service de catalogue à supprimer.

Valeur retournée :

Exemples de mode de traitement par lots

- Jacl :

```
$AdminTask modifyXSDomain {-name TestDomain -default true -modifyEndpoints
{{xhost1.ibm.com "" ,2809}} -addEndpoints {{xhost2.ibm.com "" ,2809}}}
-removeEndpoints {{xhost3.ibm.com}}}
```

- A l'aide de la chaîne Jython :

```
AdminTask.modifyXSDomain('[-name TestDomain
-default false -modifyEndpoints [[xhost1.ibm.com "" ,2809]]
-addEndpoints [[xhost3.ibm.com "" ,2809]]
-removeEndpoints [[xhost2.ibm.com]]]')
```

- Modifiez un domaine de service de catalogue existant pour activer IBM eXtremeIO :

```
AdminTask.modifyXSDomain('[-name testDomain -enableXIO true]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask modifyXSDomain {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.modifyXSDomain ('[-interactive]')
```

getTransport

La commande **getTransport** affiche le type de transport pour le domaine de service de catalogue : IBM eXtremeIO (XIO) ou ORB (Object Request Broker). Si vous exécutez cette commande sur un domaine de service de catalogue qui contient des serveurs distants ou que catalogServerName est un serveur distant, une erreur se produit. Vous devez utiliser la commande **xscmd -c showTransport** pour les serveurs distants.

Paramètres requis :

-domainName

Spécifie le nom du domaine de service de catalogue pour lequel vous voulez afficher le type de transport.

-catalogServerName

Spécifie le nom du serveur de catalogue pour lequel vous voulez afficher le type de transport.

Valeur de retour : ORB ou XIO

Affichage du type de transport d'un domaine de service de catalogue

- Jacl :

```
$AdminTask getTransport {-domainName TestDomain }
```

- A l'aide de la chaîne Jython :

```
AdminTask.getTransport('[-domainName testDomain]')
```

Affichage du transport d'un serveur de catalogue

-
- Jacl :

```
$AdminTask getTransport {-catalogServerName myCell01\myNode01\container1 }
```

- A l'aide de la chaîne Jython :

```
AdminTask.getTransport('[-catalogServerName myCell01\myNode01\container1]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask getTransport {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.getTransport ('[-interactive]')
```

testXSDomainConnection

La commande **testXSDomainConnection** teste la connexion à un domaine de service de catalogue.

Paramètres requis :

-name

Spécifie le nom du domaine de service de catalogue vers lequel tester la connexion.

Paramètres facultatifs

-timeout

Spécifie en secondes pendant combien de temps au maximum attendre la connexion.

Valeur retournée : `true` s'il est possible d'établir une connexion, sinon, une erreur de connexion est retournée.

Exemples de mode de traitement par lots

- Jacl :

```
$Admintask testXSDomainConnection
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSDomainConnection
```

Exemples de mode interactif

- Jacl :

```
$AdminTask testXSDomainConnection {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSDomainConnection ('[-interactive]')
```

testXSServerConnection

La commande **testXSServerConnection** teste la connexion à un serveur de catalogue. Cette commande fonctionne aussi bien pour les serveurs autonomes que pour les serveurs qui font partie d'un domaine de service de catalogue.

Paramètres requis :

host

Spécifie l'hôte sur lequel réside le serveur de catalogue.

listenerPort

Spécifie le port d'écoute du serveur de catalogue.

Paramètres facultatifs

timeout

Spécifie en secondes pendant combien de temps au maximum attendre la connexion au serveur de catalogue.

Valeur retournée :

Exemples de mode de traitement par lots

- Jacl :

```
$Admintask testXSServerConnection {-host xhost1.ibm.com -listenerPort 2809}
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
```

Exemples de mode interactif

- Jacl :

```
$AdminTask testXSServerConnection {-interactive}
```

- A l'aide de la chaîne Jython :

```
AdminTask.testXSServerConnection ('[-interactive]')
```

Rubrique parent : Java Création de domaines de service de catalogue dans WebSphere Application Server

Java

Configuration d'un cache local pour la mémoire cache dynamique

2.5+ Vous pouvez configurer un cache local afin d'utiliser une grille de données de mémoire cache dynamique sur le dispositif ou la collectivité. Le cache local utilise des ressources JVM locales. Généralement, le cache local comporte un sous-ensemble des données qui figurent dans la grille de données de cache dynamique sur le dispositif.

Avant de commencer

Créez une grille de données de mémoire cache dynamique. Pour plus d'informations, voir Création de grilles de données en cache dynamique.

Pourquoi et quand exécuter cette tâche

Pour activer le cache local d'une grille de données de mémoire cache dynamique, vous devez définir une propriété personnalisée pour définir le nom du modèle de mappe pour l'instance de cache dynamique WebSphere Application Server.

Procédure

Définissez la propriété personnalisée suivante pour activer le cache local :

- com.ibm.websphere.xs.dynacache.map_template_name : Indiquez `IBM_DC_NCI_PARTITIONED_.*` pour modifier le nom de modèle. La définition de cette propriété personnalisée pour une instance de cache active un cache local pour cette instance de cache.

Pour plus d'informations sur la définition de propriétés personnalisées pour la mémoire cache dynamique, voir Personnalisation d'une instance de cache dynamique avec des propriétés personnalisées.

Que faire ensuite

Par défaut, la taille maximum du cache local dynamique correspond au nombre maximum d'entrées pour une seule partition dans la grille de données du cache dynamique distant. Un algorithme d'expulsion LRU (least recently used) est utilisé pour gérer la taille. Pour mieux contrôler la taille du cache local, vous pouvez effectuer les actions suivantes :

- Configurez la propriété personnalisée com.ibm.websphere dynacache.near_cache_size pour spécifier le nombre maximum d'entrées de cache admises dans le cache local. Pour plus d'informations, voir Propriétés personnalisées de cache dynamique.

Rubrique parent : Java Création de grilles de données en cache dynamique

Configuration de la capacité maximale d'une grille de données

Vous pouvez définir une capacité maximale pour chaque grille de données de la collectivité. La configuration d'une capacité maximale limite la quantité de stockage de données qui peut être utilisée par une grille de données. La limite de capacité permet de s'assurer que la capacité de stockage disponible pour la collectivité est utilisée de façon prévisible.

Avant de commencer

- Créez les grilles de données pour votre configuration. Par défaut, les grilles de données ne sont pas configurées avec une limite de capacité maximale. Vous pouvez configurer une capacité maximale pour tous les types de grilles de données : grilles de données simples, grilles de données de session ou grilles de données de mémoire cache dynamique.

Pourquoi et quand exécuter cette tâche

Après la configuration de limites de capacité maximale sur chaque grille de données de la collectivité, la limite de capacité est appliquée en comparant la taille totale de toutes les données principales de la grille de données à la limite de capacité configurée pour la grille de données. La capacité utilisée par les copies de réplique des données n'est pas comptée lorsque la grille de données est mesurée par rapport à la limite de capacité configurée.

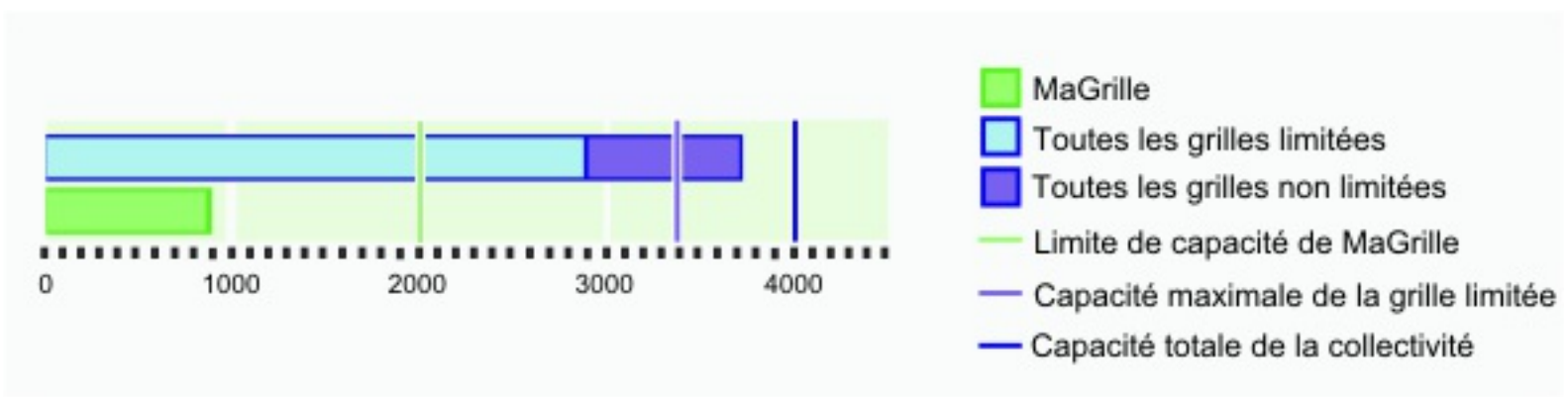
La limite de capacité maximale est une quantité maximale de données qui peut être insérée dans la grille de données. La limite ne constitue pas la garantie d'une quantité d'espace alloué pour la grille de données. Par conséquent, une grille de données peut ne pas atteindre sa limite de capacité configurée si la collectivité n'a pas la capacité pour stocker les données. Les raisons d'une capacité insuffisante dans la collectivité peuvent être une limite de capacité élevée sur la grille de données ou une capacité qui est consommée par d'autres grilles de données de la collectivité.

Lorsque la limite de capacité pour une grille de données particulière est dépassée, la grille traite les opérations d'insertion de l'une des manières suivantes :

- Par défaut, les opérations d'insertion faisant dépasser la capacité limite de la grille sont rejetées. Les processus client reçoivent une exception en réponse aux demandes d'insertion. Les opérations de lecture, de mise à jour et de suppression réussissent même si la grille de données dépasse sa limite de capacité. Avec ces opérations limitées, la grille de données peut s'exécuter à un niveau fonctionnel minimal, mais un accroissement supplémentaire de la grille de données est interdit.
- Disponible uniquement sur une grille de données simple, vous pouvez remplacer ce comportement par défaut en sélectionnant l'option **Expulsion la moins récemment utilisée pour cette grille de données**. Le fait de sélectionner cette option permet de nouvelles insertions dans une grille de données simple et maintient la limite de capacité en supprimant l'entrée de données la moins récemment utilisée. La définition de la capacité maximale sur une grille de données ne nécessite pas de redémarrage ; cependant, si vous avez sélectionné l'option d'expulsion la moins récemment utilisée sur une grille simple, cette dernière est automatiquement redémarrée pour que les modifications prennent effet. Il en est de même si vous décidez de désactiver l'option d'expulsion la moins récemment utilisée sur une grille simple.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > ***type_grille_de_données*** > ***nom_grille_de_données*** > **Afficher les attributs avancés**.
2. Sélectionnez **Limiter la quantité de capacité pour cette grille de données**.
3. Si vous définissez la capacité maximale pour une grille de données simplet et que vous voulez que la grille accepte de nouvelles opérations d'insertion (au lieu de les rejeter) aux dépens des entrées de données les moins récemment utilisées, sélectionnez l'option **Expulsion la moins récemment utilisée**. Cliquez sur **Appliquer les modifications** pour enregistrer les modifications. Vous êtes prévenu que les données de la grille seront perdues pour que le redémarrage aboutisse.
4. Affichez la consommation de capacité actuelle pour déterminer la capacité maximale à définir pour la grille de données sélectionnée. Vous pouvez aussi vous assurer que vous ne dépassez pas la capacité totale de la collectivité.



Dans ce graphique, la grille de données à configurer, MaGrille, utilise actuellement 900 Mo de capacité. Elle a une limite de capacité actuellement configurée à 2000 Mo. Au niveau de la collectivité, la capacité totale est de 4000 Mo. En outre, le total des toutes les limites configurées sur les grilles de données limitées en capacité est de 3400 Mo. Ces grilles utilisent actuellement 2900 Mo. Enfin, au moins une des grille de données de la collectivité n'a pas de limite de capacité définie. Ces grilles de données sans limite de capacité définie consomment environ 900 Mo.

5. Entrez une valeur pour la limite de consommation des données principales en Mo. Lorsque vous appuyez sur Entrée, la consommation de capacité maximale potentielle des données principales et des données répliquées s'affiche. Ce nombre varie en fonction du nombre de répliques que vous avez définies. Rappelez-vous cependant que le nombre de répliques est limité par le nombre de dispositifs de la collectivité. Si vous avez quatre répliques définies et qu'il y a trois dispositifs dans la collectivité, votre collectivité comprend une entité de données principales et deux répliques.
6. Cliquez sur **Appliquer les modifications** pour sauvegarder la configuration. Il n'est pas nécessaire de redémarrer votre grille de données pour activer la nouvelle limite.

Exemple

Exemple de limite de capacité : Plusieurs grilles de données

Des grilles de données A, B et C sont définies dans une collectivité avec une capacité de stockage totale de 600 Go. Aucune réplique n'est définie sur les grilles de données. La grille de données A a une limite de capacité de 100 Go. La grille de données B a une limite de capacité de 50 Go. La grille de données C a une limite de capacité de 200 Go. Dans ce scénario, au moins 250 Go de capacité non utilisée sont toujours disponibles dans la collectivité. La taille totale des trois grilles de données ne peut pas croître au-delà de 350 Go.

Exemple de limite de capacité : Répliques

La grille de données A est définie dans une collectivité de deux dispositifs. La grille de données A a une réplique synchrone et deux répliques asynchrones, pour un total de trois répliques. La limite de capacité de la grille est définie à 100 Mo. A l'origine, la consommation de capacité maximale de cette grille est de 200 Mo. La collectivité ayant seulement deux dispositifs, il n'existe qu'une copie principale et une copie de réplique. La grille de données principale peut utiliser jusqu'à 100 Mo. La réplique grandit dans les mêmes proportions que la grille de données principale, ce qui aboutit à une capacité consommée totale maximale de 200 Mo. Si un troisième dispositif est ajouté à la collectivité, une seconde copie de réplique est mise en place. La consommation maximale de la grille passe à 300 Mo, pour la grille principale plus les deux répliques.

Exemple de limite de capacité : Grilles de données sans limite de capacité

Des grilles de données A, B et C sont définies dans une collectivité avec une capacité de stockage totale de 600 Go. La grille de données A a une limite de capacité de 100 Go. La grille de données B a une limite de capacité de 50 Go. La grille de données C n'a pas de limite de capacité. Aucune réplique n'est définie pour aucune des trois grilles. La grille de données C n'ayant pas de limite, la grille de données peut potentiellement consommer la totalité des 600 Go de capacité disponible. Par conséquent, la grille de données A et la grille de données B ne pourraient plus insérer de données. Les données insérées par la grille de données A ou la grille de données B sont conservées, mais il n'est pas garanti que les grilles de données puissent atteindre leur limite de capacité. La grille de données C est sûre d'avoir au moins 450 Go disponibles à consommer car les seules autres grilles de données du système ne peuvent pas consommer plus d'un total de 150 Go sur les 600 Go de capacité. Ce calcul de 450 Go ignore les capacités qui sont consommées par les données répliquées. Si deux grilles de données non limitées ou plus existent dans la collectivité, la capacité potentielle d'une grille de données spécifique n'est pas garantie.

Rubrique parent : Configuration des grilles de données

Effacement des grilles de données

Vous pouvez supprimer définitivement toutes les entrées d'une grille de données. Vous pouvez effacer la grille de données pour supprimer les informations périmées ou tester les entrées.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > ***data_grid_type*** > ***data_grid_name***.
2. Cliquez sur l'icône d'effacement de grille () pour supprimer toutes les entrées de grille de données. Vous devez confirmer la suppression de toutes les entrées dans la grille de données.
3. Vous pouvez vérifier que les entrées de grille de données ont été supprimées de l'interface utilisateur. Cliquez sur **Surveiller** > **Vue d'ensemble des domaines de grilles de données** > ***data_grid_name*** et affichez le graphique **Capacités utilisées vs. Nombre d'entrées en cache**. Pour les grilles de données simples et les grilles de données de session, le nombre d'entrées dans la grille de données doit être proche de zéro. Toutefois, avec une grille de données de cache dynamique, plusieurs entrées restent dans la grille de données. Ces entrées de grille de données contiennent les informations de configuration de la grille de données de cache dynamique.

Rubrique parent : Configuration des grilles de données

Suppression des grilles de données

Si vous souhaitez supprimer les données d'une grille de données, vous pouvez supprimer la grille de données puis la recréer.

Pourquoi et quand exécuter cette tâche

Procédure

1. Dans l'interface utilisateur, cliquez sur **Grille de données** > ***type_grille_données***. Sélectionnez le nom ***nom_grille_données*** à supprimer.
2. Cliquez sur l'icône de suppression ( ) pour lancer la procédure de suppression. Un message s'affiche alors pour vous inviter à confirmer la suppression définitive de cette grille de données. Cliquez sur **OK** pour confirmer la suppression.
3. Vous pouvez contrôler la suppression de la grille de données dans la vue **Tâches**.

Résultats

Rubrique parent : Configuration des grilles de données

Java

Configuration d'un fournisseur de cache Spring

Spring Framework Version 3.1 a introduit une nouvelle abstraction de cache. Celle-ci vous permet d'ajouter de manière transparente la mise en cache à une application Spring existante. Vous pouvez utiliser WebSphere DataPower XC10 Appliance comme fournisseur de cache pour l'abstraction de cache.

Avant de commencer

- Vous devez disposer d'une application utilisant Spring Framework version 3.1 ou ultérieure.
- Votre application doit déclarer les méthodes de mise en cache à l'aide d'annotations. Pour plus d'informations sur la mise à jour de votre application pour l'abstraction de cache, voir Spring Framework Reference Documentation : Cache abstraction.
- Assurez-vous que le fichier `ogclient.jar` se trouve dans le chemin d'accès aux classes de l'application Spring.
- Si la machine virtuelle Java (JVM) sur laquelle s'exécute votre application n'est pas celle installée par WebSphere eXtreme Scale Client, vous devez ajouter l'argument JVM suivant de sorte que l'IBM Object Request Broker (ORB) soit utilisé :

  ```
  -Djava.endorsed.dirs=wxs_root/lib/endorsed
  ```

- Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.
- Lors de la connexion de l'application Spring à des grilles de données sécurisées, vous devez spécifier un fichier `client.properties` approprié comme valeur du paramètre **`client-security-config`**. Spécifiez ce paramètre dans l'ObjectGridCatalogServiceDomainBean de la configuration de conteneur Spring Inversion of Control (IoC). Vous pouvez configurer le fournisseur de cache Spring de sorte qu'il utilise l'authentification client ainsi que TLS pour le transport de réseau sécurisé. Pour plus d'informations, voir Fichier de propriétés du client, Configuration d'une application de grille de données pour l'utilisation de l'authentification client et Configuration de TLS pour les applications de grille de données.
- Vous devez connaître le nom d'hôte du serveur de catalogue et le port du dispositif. Pour obtenir ces informations, cliquez sur **Collectivité** > **Membres** dans l'interface utilisateur.

Pourquoi et quand exécuter cette tâche

A l'aide de l'abstraction de cache dans Spring Framework, vous pouvez réduire le nombre d'exécutions de la méthode d'application Spring. Lorsqu'elle est configurée, les résultats d'une méthode particulière sont placés dans la mémoire cache. Quand la méthode est ré-exécutée, l'abstraction vérifie la mémoire cache pour déterminer si les résultats de la méthode y figurent déjà. Si les résultats sont dans le cache, les résultats sont renvoyés à partir de la mémoire cache et la méthode ne s'exécute pas. La mise en oeuvre de l'abstraction peut donc réduire le nombre d'exécutions des méthodes coûteuses, ce qui réduit également le temps de réponse moyen de l'application.

Procédure

Configurez le conteneur Spring IoC (Inversion of Control) de sorte qu'il utilise WebSphere DataPower XC10 Appliance comme fournisseur de cache. L'implémentation de cache WebSphere DataPower XC10 Appliance réside dans le package com.ibm.websphere.objectgrid.spring. Définissez les beans ci-dessous dans votre configuration de conteneur Spring IoC.

```
<bean id="wxsCSDomain"
class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean"
  p:catalog-service-endpoints="CATALOG_SERVICE_ENDPOINTS"
  />

<bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
  p:object-grid-name="OBJECT_GRID_NAME"
  p:catalog-service-domain-ref="wxsCSDomain" />

<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
  <property name="caches">
    <set>
      <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
         p:name="CACHE_NAME"
         p:map-name="MAP_NAME "
                p:object-grid-client-ref="wxsGridClient" />
     </set>
   </property>
```

```
</bean>
```

CATALOG_SERVICE_ENDPOINTS

Indique le nom d'hôte et le port du serveur de catalogue.
Indique le chemin d'accès absolu ou relatif à un fichier XML ObjectGrid dans lequel modifier les paramètres côté client sous la forme d'une ressource Spring. Pour plus d'informations sur la spécification des ressources dans Spring, voir Spring Framework Reference Documentation: Resources.

Exemple : `p:client-override-xml="file:/path/to/objectgrid.xml"`

Exemple : `p:client-override-xml="classpath:com/example/app/override-objectgrid.xml"`

Exemple : `p:client-override-xml="http://myserver/override-objectgrid.xml"`

Exemple : `p:client-override-xml="ftp://myserver/override-objectgrid.xml"`

CLIENT_SECURITY_CONFIG **(facultatif)**

Indique le chemin d'accès absolu ou relatif à un fichier `client.properties` sous la forme d'une ressource Spring. Pour plus d'informations sur la spécification des ressources dans Spring, voir Spring Framework Reference Documentation: Resources. Pour plus d'informations sur la création d'un fichier `client.properties` pour WebSphere DataPower XC10 Appliance, voir Configuration d'une application de grille de données pour l'utilisation de l'authentification client.

Exemple : `p:client-security-config="file:/path/to/client.properties"`

OBJECT_GRID_NAME

Spécifie le nom d'ObjectGrid. Ce paramètre n'est pas requis si les serveurs de conteneur sont démarrés à l'aide des fichiers de configuration XML fournis. Cette valeur correspond au nom de la grille de données simple que vous avez créée dans l'interface utilisateur.

CACHE_NAME

Indique le nom de la mémoire cache spécifiée dans l'application de mise en cache Spring.

MAP_NAME

Indique le nom de la mappe de sauvegarde pour une mémoire cache. Cette valeur correspond au nom de la grille de données simple que vous avez créée dans l'interface utilisateur. Pour utiliser un nom de mappe autre que la valeur par défaut, vous pouvez définir une mappe dynamique. Pour plus d'informations sur la création de mappes dynamiques, voir Options de configuration de mappe dynamique.

Exemple

Le fragment de code ci-dessous crée une mémoire cache intitulée `default`, hébergée par un dispositif dans `myXC10.myhost.com:2809`. Cet exemple utilise l'instance de mappe par défaut qui est nommée d'après la grille de données.

```
<bean id="wxsCSDomain"
class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean"
        p:catalog-service-endpoints ="myXC10.myhost.com:2809" />
<bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
        p:object-grid-name=”my_simple_data_grid”
  p:catalog-service-domain-ref="wxsCSDomain" />
<bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
        <property name="caches">
                <set>
                        <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
                                p:name="default"
                                p:map-name=”my_simple_data_grid”
                                p:object-grid-client-ref="wxsGridClient" />
                </set>
        </property>
</bean>
```

Rubrique parent : Configuration des grilles de données

Configuration des clients

Vous pouvez configurer certaines propriétés sur les clients en remplaçant les propriétés définies sur les serveurs. Vous pouvez remplacer ces propriétés dans le fichier des propriétés du client ou à l'aide d'un programme.

Configuration des clients Java

Vous pouvez configurer WebSphere eXtreme Scale pour l'exécuter dans un environnement autonome ou dans un environnement avec WebSphere Application Server. Pour qu'un déploiement WebSphere eXtreme Scale sélectionne les modifications de configuration dans la grille de serveurs, vous devez redémarrer les processus pour que ces modifications entrent en vigueur au lieu d'être appliquées de manière dynamique. Toutefois, côté client, vous ne pouvez pas modifier les paramètres de configuration d'une instance de client existante, mais vous pouvez créer une instance de client avec les paramètres nécessaires en utilisant un fichier XML ou à l'aide d'un programme. Lorsque vous créez un client, vous pouvez remplacer les paramètres par défaut provenant de la configuration de serveur actuelle.

.NET 2.5+ **Configuration de WebSphere eXtreme Scale Client for .NET**

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET à l'aide du fichier de propriétés du client, de la configuration XML côté serveur, ou en remplaçant à l'aide d'un programme certaines propriétés du serveur.

Java

Configuration des clients Java

Vous pouvez configurer WebSphere eXtreme Scale pour l'exécuter dans un environnement autonome ou dans un environnement avec WebSphere Application Server. Pour qu'un déploiement WebSphere eXtreme Scale sélectionne les modifications de configuration dans la grille de serveurs, vous devez redémarrer les processus pour que ces modifications entrent en vigueur au lieu d'être appliquées de manière dynamique. Toutefois, côté client, vous ne pouvez pas modifier les paramètres de configuration d'une instance de client existante, mais vous pouvez créer une instance de client avec les paramètres nécessaires en utilisant un fichier XML ou à l'aide d'un programme. Lorsque vous créez un client, vous pouvez remplacer les paramètres par défaut provenant de la configuration de serveur actuelle.

Vous pouvez configurer un client eXtreme Scale (client Java uniquement) au moyen des méthodes suivantes, chacune pouvant être effectuée avec un fichier XMLde remplacement sur le client ou à l'aide d'un programme :

- Configuration XML
- Configuration par programmation
- Configuration Spring Framework
- Désactivation du cache local

Remplacements sur le client Java
Vous pouvez configurer un client WebSphere eXtreme Scale en fonction de vos besoins en remplaçant les paramètres serveur. Vous pouvez remplacer plusieurs plug-ins et attributs.

Configuration des clients Java avec une configuration XML
Vous pouvez utiliser le fichier XML de configuration pour modifier les paramètres du client.

Configuration des clients Java à l'aide d'un programme
Vous pouvez remplacer les paramètres à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

Définition du délai d'attente pour les requêtes et les nouvelles tentatives
Vous pouvez définir des options d'optimisation afin de contrôler pendant combien de temps le code du client eXtreme Scale attend la fin de l'exécution d'une requête d'accès à la grille de données ou renouvelle ses tentatives d'exécution d'une requête de ce type.

Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session
Si des applications client utilisent la gestion de session et sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour qu'il utilise la grille de données du dispositif pour la gestion de session.

Déploiement d'une passerelle REST
Vous pouvez déployer et configurer la passerelle REST pour la grille de données dans WebSphere Application Server ou dans un serveur profil Liberty.

Rubrique parent : Configuration des clients

Java

Remplacements sur le client Java

Vous pouvez configurer un client WebSphere eXtreme Scale en fonction de vos besoins en remplaçant les paramètres serveur. Vous pouvez remplacer plusieurs plug-ins et attributs.

Pour remplacer les paramètres sur un client, vous pouvez utiliser XML ou la configuration par programmation. Pour plus d'informations sur le remplacement des paramètres client, voir Configuration des clients Java avec une configuration XML et Configuration des clients Java à l'aide d'un programme.

Vous pouvez remplacer les plug-in suivants sur un client :

- **BackingMap**
  - Plug-in Evictor
  - Plug-in MapEventListener
  - Plug-in BackingMapLifecycleListener
  - Plug-in MapSerializerPlugin
- **Attributs BackingMap**
  - Attribut numberOfBuckets

    Obsolète : La propriété est obsolète. Utilisez l'attribut nearCacheEnabled pour activer le cache local.

  - attribut timeToLive
  - attribut ttlEvictorType
  - attribut evictionTriggers
  - attribut nearCacheEnabled
  - attribut nearCacheInvalidationEnabled
  - attribut nearCacheLastAccessTTLSyncEnabled
- **ObjectGrid**
  - Plug-in TransactionCallback
  - Plug-in ObjectGridEventListener
  - Plug-in ObjectGridLifecycleListener
- **attribut ObjectGrid**
  - attribut entityMetadataXMLFile
  - attribut txTimeout
  - attribut txIsolation

Rubrique parent : Configuration des clients Java

Java

Configuration des clients Java avec une configuration XML

Vous pouvez utiliser le fichier XML de configuration pour modifier les paramètres du client.

Pourquoi et quand exécuter cette tâche

Pour modifier les paramètres d'un client WebSphere eXtreme Scale, créez un fichier XML ObjectGrid dont la structure est similaire à celle du fichier utilisé pour le serveur de conteneur.

Pour la liste des modules d'extension et des attributs que vous pouvez remplacer sur le client, voir Remplacements sur le client Java.

Procédure

1. Créez un fichier XML de configuration ObjectGrid pour le client qui ait une structure semblable à celle du fichier pour le serveur de conteneur.

Supposons que le fichier XML a été associé à un fichier XML de stratégie de déploiement et que ces fichier ont été utilisés pour démarrer un serveur de conteneur.

companyGridServerSide.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyTxCallback" />
            <bean id="ObjectGridEventListener"
                className="com.company.MyOgEventListener" />
            <backingMap name="Customer"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" nearCacheEnabled="true"
                timeToLive="1600" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor"
/>
            <bean id="MapEventListener"
                className="com.company.MyMapEventListener" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Sur un serveur de conteneur, l'instance ObjectGrid nommée CompanyGrid se comporte conformément à ce qui est défini dans le fichier `companyGridServerSide.xml`. Par défaut, les paramètres du client CompanyGrid sont identiques à ceux de l'instance CompanyGrid qui s'exécute sur le serveur.

Le fichier XML ObjectGrid suivant peut être utilisé pour définir certains attributs et plug-in du client CompanyGrid.

```
companyGridClientSide.xml

<?xml version="1.0" encoding="UTF-8"?>
```

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyClientTxCallback" />
            <bean id="ObjectGridEventListener" className="" />
            <backingMap name="Customer" nearCacheEnabled="true"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" nearCacheEnabled="true"
                timeToLive="800" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor"
/>
            <bean id="MapEventListener" className="" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Le fichier XML spécifie les remplacements suivants :

- Le bean TransactionCallback sur le client est com.company.MyClientTxCallback au lieu du paramètre côté serveur com.company.MyTxCallback.
- Le client n'est associé à aucun plug-in ObjectGridEventListener car la valeur className est la chaîne vide.
- Le client active un cache local pour Customer backingMap, conserve son plug-in Evictor et supprime le plug-in MapEventListener.
- L'attribut timeToLive d'OrderLine backingMap a changé.
- Bien qu'un attribut lockStrategy différent ait été indiqué, les conséquences sont nulles car cet attribut n'est pas pris en charge pour un remplacement par le client.

2. Créez le client avec le fichier XML.

   Pour créer le client CompanyGrid à l'aide du fichier `companyGridClientSide.xml`, transmettez le fichier XML ObjectGrid sous la forme d'une URL à l'une des méthodes de connexion dans l'interface ObjectGridManager :

   ```
   ObjectGridManager ogManager =
           ObjectGridManagerFactory.ObjectGridManager();
   ClientClusterContext clientClusterContext =
           ogManager.connect("MyServer1.company.com:2809", null, new URL(
                   "file:xml/companyGridClientSide.xml"));
   ```

Rubrique parent : Configuration des clients Java

Java

Configuration des clients Java à l'aide d'un programme

Vous pouvez remplacer les paramètres à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

Pourquoi et quand exécuter cette tâche

L'exemple de code suivant crée les mêmes substitutions que celles décrites dans Configuration des clients Java avec une configuration XML.

Pour la liste des modules d'extension et des attributs que vous pouvez remplacer sur le client, voir Remplacements sur le client Java.

Procédure

Le code suivant crée une instance ObjectGrid côté client.

```
ObjectGridConfiguration companyGridConfig = ObjectGridConfigFactory
    .createObjectGridConfiguration("CompanyGrid");
Plugin txCallbackPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.TRANSACTION_CALLBACK, "com.company.MyClientTxCallback");
companyGridConfig.addPlugin(txCallbackPlugin);

Plugin ogEventListenerPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.OBJECTGRID_EVENT_LISTENER, "");
companyGridConfig.addPlugin(ogEventListenerPlugin);

BackingMapConfiguration customerMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("Customer");
customerMapConfig.setNumberOfBuckets(1429);
Plugin evictorPlugin = ObjectGridConfigFactory.createPlugin(PluginType.EVICTOR,
    "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor");
customerMapConfig.addPlugin(evictorPlugin);

companyGridConfig.addBackingMapConfiguration(customerMapConfig);

BackingMapConfiguration orderLineMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("OrderLine");
orderLineMapConfig.setNumberOfBuckets(701);
orderLineMapConfig.setTimeToLive(800);
orderLineMapConfig.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);

companyGridConfig.addBackingMapConfiguration(orderLineMapConfig);

List ogConfigs = new ArrayList();
ogConfigs.add(companyGridConfig);

Map overrideMap = new HashMap();
overrideMap.put(CatalogServerProperties.DEFAULT_DOMAIN, ogConfigs);

ogManager.setOverrideObjectGridConfigurations(overrideMap);
ClientClusterContext client = ogManager.connect(catalogServerEndpoints, null, null);
ObjectGrid companyGrid = ogManager.getObjectGrid(client, objectGridName);
```

L'instance ogManager de l'interface ObjectGridManager recherche uniquement les remplacements dans les objets ObjectGridConfiguration et BackingMapConfiguration que vous incluez dans la mappe overrideMap. Par exemple, le code précédent remplace le nombre de compartiments de la mappe OrderLine. La mappe Order reste cependant inchangée côté client car aucune configuration de cette mappe n'est incluse.

Rubrique parent : Configuration des clients Java

Java

Définition du délai d'attente pour les requêtes et les nouvelles tentatives

Vous pouvez définir des options d'optimisation afin de contrôler pendant combien de temps le code du client eXtreme Scale attend la fin de l'exécution d'une requête d'accès à la grille de données ou renouvelle ses tentatives d'exécution d'une requête de ce type.

Pourquoi et quand exécuter cette tâche

Vous pouvez configurer des paramètres du client eXtreme Scale qui contrôlent pendant combien de temps le client tente de créer des connexions réseau, de traiter une requête de grille de données pour une partition et de renouveler cette requête avant de renvoyer une exception à votre application.

Facteurs d'optimisation du délai d'attente pour les requêtes et les nouvelles tentatives pour XIO et ORB

Pour certaines options d'optimisation, l'emplacement de définition des délais d'attente dépend du transport utilisé : eXtremeIO (XIO) ou Object Request Broker (ORB). Ces options de niveau transport sont les premières à avoir une incidence sur les interactions avec votre client, car elles déterminent pendant combien de temps le transport tente d'établir des connexions socket réseau et de combien de temps un appel de procédure éloignée (RPC, remote procedure call) analogue à une opération de grille de données dispose pour son exécution.

Lorsque vous optimisez ces valeurs, tenez compte de ce que votre environnement peut tolérer dans des conditions de charge maximum et d'état stabilisé. Si vous définissez les intervalles trop en dessous des valeurs par défaut (30 secondes pour le délai d'attente pour les nouvelles tentatives, par exemple), vos opérations risquent d'échouer prématurément. Tenez compte des facteurs suivants :

- Temps d'attente réseau
- Couplage des interactions de la grille avec des ressources externes telles que des bases de données
- Pauses de la récupération de place qui résultent de votre combinaison de politiques d'optimisation de taille de segment de mémoire, d'utilisation de segment de mémoire et de récupération de place

Paramètres ORB d'optimisation des délais d'attente pour les requêtes et les nouvelles tentatives

Pour ORB, les paramètres de délai d'attente sont les suivants :

com.ibm.CORBA.ConnectionTimeout

Délai pendant lequel ORB tente de créer une connexion socket avec l'emplacement distant. ORB place ces connexions en cache, ce qui fait que cette opération n'est pas répétée pour chaque requête.

com.ibm.CORBA.RequestTimeout

Délai pendant lequel ORB attend la fin de l'exécution d'un appel de procédure éloignée.

com.ibm.CORBA.FragmentTimeout

Pour plus de détails, voir la documentation IBM consacrée à ORB. Le produit est fourni avec une valeur par défaut pour cette valeur.

com.ibm.CORBA.LocateRequestTimeout

Pour plus de détails, voir la documentation IBM consacrée à ORB. Le produit est fourni avec une valeur par défaut pour cette valeur.

Lorsque vous optimisez les paramètres RequestTimeout et ConnectionTimeout, il peut être intéressant de les ajuster en tenant compte des recommandations par défaut. Vous pouvez également leur affecter la même valeur, basée sur le délai d'attente souhaité pour les requêtes.

Le niveau d'optimisation suivant concerne le paramètre requestRetryTimeout. Pour chaque type de transport, après qu'il a émis une exception système car un appel de procédure éloignée n'a pas abouti dans le délai imparti, la grille de données peut utiliser le délai supplémentaire défini par le paramètre requestRetryTimeout (par exemple, le délai d'attente pour les requêtes peut être de 10 secondes et le délai d'attente pour les nouvelles tentatives, de 20 secondes) pour définir pendant combien de temps elle effectue les actions suivantes :

- Envoi au serveur de catalogue d'une demande asynchrone en vue d'obtenir la table de routage la plus récente, pour le cas où les partitions seraient situées ailleurs en raison d'un basculement.
- Utilisation de nouvelles routes et nouvelle tentative d'exécution de la requête, ou arrêt des tentatives et envoi d'une exception à votre application.

La propriété requestRetryTimeout est définie en millisecondes. Une valeur supérieure à zéro indique que la demande doit être renouvelée lors de l'émission d'une exception pour laquelle une nouvelle tentative est possible. Une valeur de `0` indique qu'aucune nouvelle tentative ne doit avoir lieu en cas d'exception. Pour utiliser le comportement par défaut, supprimez la propriété ou attribuez-lui la valeur `-1`.

Paramètres XIO d'optimisation des délais d'attente pour les nouvelles tentatives

Pour XIO, les paramètres consolidés suivants sont disponibles :

- Le paramètre xioTimeout détermine le délai pendant lequel le transport XIO tente d'établir une connexion socket réseau.
- Les paramètres ORB LocateRequest et FragmentTimeout n'ont pas d'équivalent.
- Le paramètre requestRetryTimeout contrôle le délai maximum dont un appel de procédure éloignée (RPC) dispose pour aboutir, et il aide à contrôler pendant combien de temps de nouvelles tentatives d'appel RPC ont lieu lorsque de nouvelles informations de routage sont obtenues du serveur de catalogue. Lorsque vous utilisez le transport XIO, il signale souvent beaucoup plus tôt que ORB qu'un appel de procédure éloignée est sur le point d'échouer. Cet échec se produit avant l'expiration du délai d'attente requestRetryTimeout.

Comment définir le délai d'attente pour les nouvelles tentatives

Le délai d'attente pour les nouvelles tentatives d'exécution des requêtes peut être défini dans le fichier des propriétés du client ou dans une session. La valeur définie dans la session remplace la valeur qui figure dans les propriétés du client. Si la valeur définie est supérieure à zéro, la demande est renouvelée jusqu'à ce que le délai d'attente expire ou qu'une erreur permanente se produise. Une erreur permanente peut être une exception `DuplicateKeyException`. La valeur zéro définit le mode "fail-fast" et la grille de données ne retente pas la transaction, quel que soit son type.

Délai d'attente pour les transactions et délai d'attente pour les nouvelles tentatives d'exécution de requête

Pendant l'exécution, le délai d'attente pour les transactions est utilisé avec le délai d'attente pour les nouvelles tentatives d'exécution de requête, ce qui garantit que ce dernier ne dépasse pas le délai d'attente pour les transactions.

Deux types de transaction existent : les transactions à validation automatique et les transactions qui utilisent des méthodes explicites begin et commit. Les exceptions qui autorisent de nouvelles tentatives diffèrent pour ces deux types de transaction :

- Les transactions appelées dans une session sont retentées pour ORB CORBA SystemException (TransportException pour XIO) et les exceptions TargetNotAvailable du client eXtreme Scale.
- Les transactions à validation automatique sont retentées pour CORBA SystemException et les exceptions de disponibilité du client eXtreme Scale. Ces dernières sont ReplicationVotedToRollbackTransactionException, TargetNotAvailable et AvailabilityException.

Les erreurs d'application et autres erreurs permanentes provoquent l'émission immédiate d'une exception et le client ne retente pas la transaction. Ces erreurs permanentes incluent les exceptions DuplicateKeyException et KeyNotFoundException. Utilisez le paramètre "fail-fast" pour émettre toutes les exceptions sans retenter les transactions.

Exceptions pour lesquelles le client retente la transaction :

- ReplicationVotedToRollbackTransactionException (uniquement en validation automatique)
- TargetNotAvailable
- org.omg.CORBA.SystemException (TransportException est l'équivalent XIO de cette exception système ORB)
- AvailabilityException (uniquement en validation automatique)
- LockTimeoutException (uniquement en validation automatique)
- UnavailableServiceException (uniquement en validation automatique)

Exceptions permanentes pour lesquelles la transaction n'est pas retentée :

- DuplicateKeyException
- KeyNotFoundException
- LoaderException
- TransactionAffinityException
- LockDeadlockException
- OptimisticCollisionException

Procédure

- Définition du délai d'attente pour les nouvelles tentatives dans un fichier de propriétés de client.

  Pour définir la valeur de requestRetryTimeout dans un client, ajoutez ou modifiez cette propriété dans le Fichier de propriétés du client. Par défaut, le fichier dans lequel les propriétés du client sont définies est le fichier `objectGridClient.properties`. La propriété requestRetryTimeout est définie en millisecondes. Une valeur supérieure à zéro indique que la demande doit être renouvelée lors de l'émission d'une exception pour laquelle une nouvelle tentative est possible. Une valeur de `0` indique qu'aucune nouvelle tentative ne doit avoir lieu en cas d'exception. Pour utiliser le comportement par

défaut, supprimez la propriété ou attribuez-lui la valeur `-1`. Voici un exemple de valeur dans le fichier `objectGridClient.properties` :

```
requestRetryTimeout = 30000
```

La valeur de requestRetryTimeout est spécifiée en millisecondes. Dans l'exemple, si la valeur est utilisée dans une instance ObjectGrid, la valeur de requestRetryTimeout sera de 30 secondes.

- Définition du délai d'attente pour les nouvelles tentatives à l'aide d'un programme.

  Pour définir les propriétés du client à l'aide d'un programme, commencez par créer un fichier de propriétés dans un <`emplacement`> approprié pour votre application. Dans l'exemple ci-dessous, le fichier des propriétés du client est celui cité dans le fragment de code de la section précédente (`objectGridClient.properties`). Après vous être connecté à une instance ObjectGridManager, définissez les propriétés du client comme indiqué. Ensuite, lorsque vous avez une instance ObjectGrid, cette instance possède les propriétés client que vous avez définies dans le fichier. Chaque fois que vous serez amené à modifier ce fichier, vous devrez explicitement obtenir une nouvelle instance ObjectGrid.

  ```
  ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
  String objectGridName = “testObjectGrid”;
  URL clientXML = null;
  ClientClusterContext ccc = manager.connect("localhost:2809", null, clientXML);
  File file = new File("<location>/objectGridClient.properties");
  URL url = file.toURI().toURL();
  ccc.setClientProperties(objectGridName, url);
  ObjectGrid objectGrid = ogManager.getObjectGrid(ccc, objectGridName);
  ```

- Définissez le fichier de substitution pendant une validation de session.

  Pour définir dans l'objet Session pendant combien de temps il convient d'effectuer de nouvelles tentatives, ou pour remplacer la propriété client requestRetryTimeout, appelez la méthode setRequestRetryTimeout(long) dans l'interface Session.

  ```
  Session sessionA = objectGrid.getSession();
  sessionA.setRequestRetryTimeout(30000);
  ObjectMap mapA = sessionA.getMap("payroll");
  String key = "key:" + j;
  mapA.insert(key, "valueA");
  ```

  Cette session utilise à présent la valeur 30 000 millisecondes (30 secondes) pour requestRetryTimeout, quelle que soit la valeur définie dans le fichier des propriétés du client. Pour plus d'informations sur l'interface de session, voir Utilisation des sessions pour accéder aux données de la grille.

Exemple

Dans l'exemple ci-dessous, le client peut traiter le temps d'attente réseau, la récupération de place et les conflits généraux sur le serveur grâce à la définition de délais d'attente courts. La propriété **`requestRetryTimeout`** est définie sur 10 secondes, et la propriété **`xioTimeout`** est définie comme la valeur ORB **`ConnectionTimeout`**, c'est-à-dire sur 5 secondes.

Tableau 1. Configurations de grille de données pour les types de transport ORB et eXtremeIO

Type de grille	**ORB**	**XIO**
Une application client Java™ ou .NET qui accède directement à une API eXtreme Scale	• Modifiez le fichier `orb.properties` de votre application client. Définissez les valeurs suivantes : ◦ com.ibm.CORBA.RequestTimeout=5 ◦ com.ibm.CORBA.ConnectTimeout=5 ◦ com.ibm.CORBA.FragmentTimeout=5 ◦ com.ibm.CORBA.LocateRequestTimeout=5 **Remarque :** Avec WebSphere Application Server, vous contrôlez les paramètres ORB à l'aide du gestionnaire de déploiement et non pas du fichier `orb.properties`. • Modifiez le fichier `objectGridClient.properties` de	Modifiez le fichier `objectGridClient.properties` de votre application client en indiquant les valeurs suivantes : • xioRequestTimeout=50000. Cette valeur représente des millisecondes et correspond au paramètre com.ibm.CORBA.RequestTime

	`objectGridClient.properties` de l'application client en indiquant requestRetryTimeout=7000.	out. • xioTimeout=5. Cette valeur représente des secondes et correspond au paramètre com.ibm.CORBA.ConnectTimeout. • requestRetryTimeout=70000. Cette valeur représente des millisecondes et est également utilisée pour le transport ORB. • ORB FragmentTimeout et LocateRequestTimeout n'ont pas d'équivalent en XIO.
Session HTTP	Placez le fichier de propriétés du client dans le chemin d'accès aux classes, en y définissant la valeur `requestRetryTimeout=10000`.	Modifiez le fichier de propriétés de client de votre application client en indiquant les valeurs suivantes : • xioRequestTimeout=10000. Cette valeur représente des millisecondes et correspond au paramètre com.ibm.CORBA.RequestTimeout. • xioTimeout=5. Cette valeur représente des secondes et correspond au paramètre com.ibm.CORBA.ConnectTimeout. • requestRetryTimeout=10000. Cette valeur représente des millisecondes et est également utilisée pour le transport ORB. • ORB FragmentTimeout et LocateRequestTimeout n'ont pas

		d'équivalent en XIO.
Cache dynamique	Définissez la propriété suivante dans l'instance de mémoire cache : • com.ibm.websphere.xs.dynacache.request_retry_timeout_override=2000 Si vous souhaitez que la valeur de cette propriété d'instance de cache soit utilisée pour toutes les instances de mémoire cache dynamique, vous pouvez, au lieu d'utiliser cette propriété, utiliser le paramètre **requestRetryTimeout** en plaçant le fichier de propriétés du client dans le chemin d'accès aux classes.	

Rubrique parent : Configuration des clients Java

Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Si des applications client utilisent la gestion de session et sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour qu'il utilise la grille de données du dispositif pour la gestion de session.

Avant de commencer

Cette tâche inclut les instructions qui permettent de déployer une application qui requiert la gestion de session. Pour savoir comment créer une application Java qui utilise la gestion de session, voir Développement d'applications de grilles de données avec des API Java.

Pourquoi et quand exécuter cette tâche

De même que vous pouvez configurer vos applications WebSphere Application Server pour qu'elles utilisent le dispositif pour la gestion de session, vous pouvez configurer le profil Liberty à cette fin.

Vous pouvez décider d'utiliser le profil Liberty avec le dispositif si vous avez besoin d'un serveur léger offrant des fonctions dynamiques. Par exemple, vous pouvez ajouter ou supprimer des fonctions, lesquelles sont des entités qui permettent de contrôler les éléments de l'environnement d'exécution chargés sur un serveur donné. Par conséquent, dans le profil Liberty, si vous exécutez des applications qui gèrent les sessions, par exemple, vous pouvez créer une définition de serveur que vous utilisez pour spécifier des fonctions du profil Liberty, lesquelles contrôlent la façon dont le serveur interagit avec la grille de données du dispositif.

Procédure

1. Créez une définition de serveur en exécutant la commande suivante :

   ```
   répertoire_install_wlp/bin/server create nom_serveur
   ```

2. Recherchez le fichier `server.xml` sous la définition du serveur et ouvrez-le dans un éditeur XML.
3. Placez l'application de session (par exemple, *votre_application*`.jar`) dans le répertoire `/wlp/usr/servers/defaultServer/apps`.
4. Démarrez la console de surveillance du dispositif, puis cliquez sur **Grille de données** > **Grille de session**.
5. Créez sur le dispositif une grille de session nommée `session`.
6. Exportez la propriété **`JAVA_HOME`** à partir d'une ligne de commande. Veillez à exécuter la commande à partir du répertoire dans lequel est installé le profil Liberty.
7. Démarrez le profil Liberty à l'aide de la commande suivante :

   ```
   ./server start servername
   ```

   Un PID s'affiche.
8. Ouvrez l'application de session à l'aide de l'URL suivante :

   ```
   http://server:port/A/
   ```

9. Exécutez des tests de session pour vérifier que les données sont bien écrites dans la grille de session du dispositif.

Fichier de définition de serveur sans SSL activé

Voici l'exemple d'un fichier `server.xml` de base pour lequel SSL n'est pas activé. L'exemple suivant est présenté sur plusieurs lignes en raison des contraintes liées à la publication.

Remarque : La fonction Web est obsolète. Utilisez la fonction webApp à la place. Lorsque vous ajoutez la fonction Web à la définition de serveur et configurez le gestionnaire de sessions, vous pouvez utiliser la réplication de session dans les applications WebSphere® eXtreme Scale exécutées dans le profil Liberty.

Etudiez l'exemple suivant, dans lequel la fonction webApp est utilisée :

2.5+

```
<server description="new server">

    <!-- Enable features -->
```

```
    <featureManager>
        <feature>jsp-2.2</feature>
      <feature>eXtremeScale.server-1.1</feature>
      <feature>eXtremeScale.webApp-1.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />

    <xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809"
securityEnabled="false" />
    <applicationMonitor updateTrigger="mbean"/>
    <application id="A" location="A.ear" name="A" type="ear"/>
     <httpSession cloneId="A_test"/>

   <!-- <com.ibm.ws.xs.server.config catalogServer="true" /> -->
</server>
```

Configuration du profil Liberty pour les grilles de données qui font l'objet d'un accès avec SSL activé
Si des applications client sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour HTTPS, lequel utilise automatiquement SSL et le chiffrement de données pour les serveurs Web sécurisés.

Configuration du profil Liberty pour une exécution avec des clients
Utilisez la fonction du client WebSphere eXtreme Scale pour exécuter le profil Liberty avec des clients eXtreme Scale.

Activation de la fonction webApp dans le profil Liberty
Un serveur de profil Liberty peut héberger une grille de données qui place en mémoire cache les données des applications en vue de la réplication des données de session HTTP pour la tolérance aux pannes.

Propriétés de la fonction xsDynacacheApp du profil Liberty
Spécifiez le profil Liberty qui doit héberger la grille de données, que vous pouvez configurer comme fournisseur de cache dynamique à l'aide de cette fonction.

Rubrique parent : Configuration des clients Java

Configuration du profil Liberty pour les grilles de données qui font l'objet d'un accès avec SSL activé

Si des applications client sont déployées dans le profil Liberty de WebSphere Application Server, vous pouvez configurer le profil Liberty pour HTTPS, lequel utilise automatiquement SSL et le chiffrement de données pour les serveurs Web sécurisés.

Procédure

1. Exécutez la commande suivante pour créer le certificat SSL et activer le protocole HTTPS :

   ```
   cd to lib_dir\bin
   securityUtility createSSLCertificate --server=defaultServer --password=xc10test
   ```

2. Ajoutez la fonction SSL suivante au fichier `server.xml` afin de configurer le profil Liberty pour une exécution du chiffrement de données SSL :
   2.5+

   ```
   <featureManager>
           <feature>ssl-1.1</feature>
       </featureManager>
       <keyStore id="defaultKeyStore" password="{xor}MjowbTI+Kyw=" />
   ```

3. Démarrez le profil Liberty à l'aide de la commande suivante :

   ```
   ./server start servername
   ```

   Un PID s'affiche.

4. Ouvrez l'application de session à l'aide de l'URL suivante :

   ```
   http://server:securedport/A/
   ```

5. Exécutez des tests de session pour vérifier que les données sont écrites dans la grille de session du dispositif.

Exemple de fichier de définition activé par SSL

Certaines lignes de code sont réparties sur plusieurs lignes pour des raisons de mise en page. Voici l'exemple d'une configuration de fichier `server.xml` avancée qui utilise la fonction SSL. L'exemple suivant est présenté sur plusieurs lignes en raison des contraintes de mise en page.

Remarque : A compter de la version 2.5, le numéro de version de fonction `webApp-1.0` devient `webApp-1.1`.

2.5+

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>eXtremeScale.server-1.1</feature>
      <feature>eXtremeScale.webApp-1.1</feature>
       <feature>ssl-1.1</feature>
    </featureManager>

      <httpEndpoint id="defaultHttpEndpoint"
                    host="*"
                    httpPort="9080"
                    httpsPort="9443">
        <!--tcpOptions soReuseAddr="true" / -->
    </httpEndpoint>

    <keyStore id="defaultKeyStore" password="{xor}Jzxubys6LCs=" />

 <xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809"
securityEnabled="true"
credentialGeneratorClass="com.ibm.websphere.objectgrid.security.plugins.builtins.UserPassw
ordCredentialGenerator"
```

```
   credentialGeneratorProps="xcadmin xcadmin"/>
    <applicationMonitor updateTrigger="mbean"/>
    <application id="A" location="A.ear" name="A" type="ear"/>
    <httpSession cloneId="A_test"/>

</server>
```

Que faire ensuite

Pour définir la configuration SSL entre le profil Liberty et le conteneur de la grille de données, spécifiez le type de transport du client dans le fichier de propriétés du client. Les valeurs possibles sont :

- **`TCP/IP`** : Indique que le client ne prend en charge que les connexions TCP/IP.
- **`SSL pris en charge`** : Indique que le client prend en charge les connexions TCP/IP et SSL (Secure Sockets Layer). (Valeur par défaut)
- **`SSL requis`**: Indique que le client exige des connexions SSL.

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Tâches associées:
Configuration de la sécurité du client

Configuration du profil Liberty pour une exécution avec des clients

Utilisez la fonction du client WebSphere eXtreme Scale pour exécuter le profil Liberty avec des clients eXtreme Scale.

Avant de commencer

Effectuez les tâches suivantes avant de configurer le profil Liberty :

- Installez le profil Liberty et WebSphere eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Cette configuration ne fournit que la fonction client. La fonction serveur s'exécute dans un autre processus. L'ajout de la fonction client permet à l'application d'accéder aux API eXtreme Scale et de se connecter à une grille distante.

Cette configuration client fournit un processus unique incluant ce dont vous avez besoin pour exécuter une application Web à l'aide d'une grille de données eXtreme Scale. Après avoir ajouté la fonction client, l'application peut écrire dans les API eXtreme Scale.

Procédure

N'ajoutez la fonction client qu'au serveur du profil Liberty. Ajoutez le code suivant au serveur du profil Liberty :

```
eXtremeScale.client-1.1
```

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Activation de la fonction webApp dans le profil Liberty

Un serveur de profil Liberty peut héberger une grille de données qui place en mémoire cache les données des applications en vue de la réplication des données de session HTTP pour la tolérance aux pannes.

Pourquoi et quand exécuter cette tâche

Lorsque vous installez le WebSphere Application Server profil Liberty, il ne contient pas la réplication de session. Cependant, si vous utilisez la grille de données avec le profil Liberty, vous pouvez répliquer les sessions pour que les utilisateurs de l'application ne perdent pas les données de session en cas de défaillance d'un serveur.

Lorsque vous ajoutez la fonction Web à la définition de serveur et configurez le gestionnaire de sessions, vous pouvez utiliser la réplication de session dans les applications de grille de données exécutées dans le profil Liberty.

Procédure

Ajoutez la fonction webApp ci-dessous au fichier profil Liberty `server.xml`. La fonction webApp inclut la fonction client, mais pas la fonction serveur. Vous voudrez certainement séparer les applications Web des grilles de données. Par exemple, vous disposez d'un serveur profil Liberty pour vos applications Web et d'un autre serveur profil Liberty pour l'hébergement de la grille de données.

```
<featureManager>
<feature>eXtremeScale_webapp-1.1</feature>
</featureManager>
```

Résultats

Vos applications Web peuvent maintenant conserver leurs données de session dans une grille de données.

Exemple

Etudiez l'exemple de fichier `server.xml` suivant, qui contient la fonction Web que vous utilisez lorsque vous vous connectez à la grille de données à distance :

```
<server description="Airport Entry eXtremeScale Getting Started Client Web Server">
<!--
Ce programme exemple n'est soumis à aucune redevance ; il est fourni EN L'ETAT et peut
être librement utilisé, exécuté, copié et modifié
sans paiement de redevance par le client
(a) pour sa propre formation,
(b) pour développer des applications qui doivent s'exécuter avec un produit IBM WebSphere,
à des fins d'utilisation interne par le client pour que le client le redistribue dans le
cadre d'une telle application
dans les propres produits du client.
Licensed Materials - Property of IBM
5724-X67, 5655-V66 (C) COPYRIGHT International Business Machines Corp. 2012
-->
<!-- Enable features -->
<featureManager>
<feature>eXtremeScale.webapp-1.1</feature>
</featureManager>

<httpEndpoint id="defaultHttpEndpoint"
host="*"
httpPort="${default.http.port}"
httpsPort="${default.https.port}" />

<xsWebApp objectGridName="session" catalogHostPort="remoteHost:2809"
securityEnabled="false" />

</server>
```

Que faire ensuite

La fonction webApp dispose de propriétés de métadonnées que vous pouvez définir dans l'élément xsWebApp du fichier `server.xml`. Pour plus d'informations, voir Propriétés de la fonction xsWebApp du profil Liberty.

[Propriétés de la fonction xsWebApp du profil Liberty]
Définissez la fonction webApp pour étendre l'application Web dans le profil Liberty. Ajoutez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Propriétés de la fonction xsWebApp du profil Liberty

Définissez la fonction webApp pour étendre l'application Web dans le profil Liberty. Ajoutez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

La fonction Web est obsolète. Utilisez la fonction webApp lorsque vous voulez répliquer les données de session HTTP pour la tolérance aux pannes.

Vous pouvez définir les attributs suivants dans l'élément xsWebApp du fichier `server.xml` :

Paramètres

`objectGridName`

Valeur de chaîne qui définit le nom de l'instance ObjectGrid utilisée pour une application Web particulière. Le nom par défaut est session.

Cette propriété doit refléter le nom objectGridName dans les fichiers XML ObjectGrid et XML de déploiement utilisés pour démarrer les serveurs de conteneur eXtreme Scale.

`catalogHostPort`

Le serveur de catalogues peut être contacté pour obtenir une instance ObjectGrid côté client. La valeur doit avoir le format `host:port<,hôte:port>`. L'hôte est le programme d'écoute sur lequel le serveur de catalogue s'exécute. Le port est le port d'écoute du processus serveur de catalogue. La longueur de cette liste peut être arbitraire et la liste n'est utilisée que pour l'amorçage. La première adresse viable qui est utilisée. Elle est facultative dans WebSphere Application Server si la propriété **`catalog.services.cluster`** est définie.

`replicationInterval`

Entier (en secondes) qui définit le temps séparant deux écritures de sessions actualisées vers la grille. La valeur par défaut est 10 secondes. Les valeurs possibles sont comprises entre 0 et 60. 0 signifie que les sessions actualisées sont écrites dans la grille pour chaque demande dès la fin de l'appel à la méthode de service du servlet. Une valeur `replicationInterval` plus élevée améliore les performances, car un moins grand nombre de mises à jour sont écrites dans la grille de données. Mais en même temps, une valeur supérieure à 0 rend la configuration moins tolérante aux pannes.

Ce paramètre s'applique uniquement lorsque objectGridType a la valeur `REMOTE`.

`sessionTableSize`

Entier qui définit le nombre de références de session conservées en mémoire. La valeur par défaut est `1000`.

Ce paramètre appartient uniquement à une topologie `REMOTE`, car la topologie `EMBEDDED` a déjà les données de session dans le même groupe que le conteneur Web.

Les sessions sont expulsées de la table interne en fonction de la logique LRU (least recently used). Lorsqu'une session est expulsée de cette table, elle est invalidée dans le conteneur Web. Cependant, les données ne sont pas pour autant supprimées de la grille, ce qui permet aux demandes ultérieures de cette session de continuer à extraire les données. Cette valeur doit être supérieure à la valeur maximale du pool d'unités d'exécution du conteneur Web, ce qui réduit les conflits au niveau du cache de session.

`fragmentedSession`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `true`. Ce paramètre permet de contrôler si le produit stocke les données de session en tant qu'entrée entière ou s'il stocke chaque attribut séparément.

Affectez au paramètre fragmentedSession la valeur `true` si la session d'application Web a de nombreux attributs ou des attributs avec des grandes tailles. Affectez à fragmentedSession la valeur `false` si une session a peu d'attributs, car tous les attributs sont stockés dans la même clé dans la grille de données.

Dans la précédente implémentation à base de filtres, il était fait référence à cette propriété en tant que mécanisme de persistance avec, comme valeurs possibles, ObjectGridStore (fragmentation) et ObjectGridAtomicSessionStore (non-fragmentation).

`securityEnabled`

Valeur de type chaîne `true` ou `false`. La valeur par défaut est `false`. Ce paramètre active la sécurité du client eXtreme Scale. Il doit correspondre au paramètre **`securityEnabled`** dans le fichier des propriétés sur serveur eXtreme Scale. Si les paramètres ne correspondent pas, une exception est générée.

credentialAuthentication

Indique si l'authentification des données d'identification est imposée ou prise en charge.

Jamais

Aucune authentification de certificat client n'est imposée.

Requise

L'authentification des données d'identification est toujours appliquée. Si le serveur ne prend pas en charge l'authentification des données d'identification, le client ne peut pas se connecter au serveur.

Pris en charge

(Par défaut) L'authentification des données d'identification est imposée seulement si à la fois le client et le serveur prennent en charge l'authentification des données d'identification.

authenticationRetryCount

Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe utilisée pour obtenir les données d'identification des clients. La valeur par défaut est `0`.

credentialGeneratorClass

Le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe sert à obtenir les données d'identification des clients.

credentialGeneratorProps

Les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés correspondent à l'objet avec la méthode setProperties(String). La valeur credentialGeneratorProps n'est utilisée que si la valeur de la propriété **credentialGeneratorClass** n'est pas null.

shareSessionsAcrossWebApps

Spécifie si les sessions sont partagées entre des applications Web ; spécifiée comme valeur de chaîne `true` ou `false`. La valeur par défaut est `false`. La spécification de servlet indique que les sessions HTTP ne peuvent pas être partagées entre des applications Web. Une extension à la spécification de servlet est fournie pour permettre ce partage.

Rubrique parent : Activation de la fonction webApp dans le profil Liberty

Propriétés de la fonction xsDynacacheApp du profil Liberty

Spécifiez le profil Liberty qui doit héberger la grille de données, que vous pouvez configurer comme fournisseur de cache dynamique à l'aide de cette fonction.

Vous pouvez définir les attributs suivants dans l'élément `xsDynacacheApp` du fichier `server.xml` :

Paramètres

`gridName`

Valeur de chaîne qui définit le nom de l'instance de grille de données utilisée pour une instance de cache dynamique particulière.

`cacheName`

Définit le nom du suffixe unique qui est utilisé comme nom du modèle de mappe. Par exemple :

```
IBM_DC_PARTITIONED.nom_cache
```

`mapName`

Définit le nom de la mappe de la grille de données qui joue le rôle de fournisseur de cache dynamique.

`remoteDomain`

Définit le nom de domaine client éloigné du fournisseur de cache dynamique de la grille de données dans le profil Liberty.

`clientObjectgridXML`

Définit l'emplacement du fichier `objectgrid.xml` client pour eXtreme Scale.

`requestRetryTimeout`

Définit le temps (en millisecondes) pendant lequel une requête peut s'exécuter avant qu'un dépassement de délai ne se produise. Cette propriété remplace la propriété **`requestRetryTimeout`** si elle est définie dans le fichier de propriétés du client. La valeur par défaut pour les instances de mémoire cache dynamique est de `2000` millisecondes.

`templateName`

Indique le nom du préfixe du modèle de mappe. Vous pouvez utiliser l'un des modèles de mappe suivants :

- `IBM_DC_PARTITIONED_.*` (valeur par défaut)
- `IBM_DC_NCI_PARTITIONED_.*` (indique que ce cache utilise un cache local)

Rubrique parent : Configuration du profil Liberty pour qu'il utilise la grille de données pour la gestion de session

Déploiement d'une passerelle REST

Vous pouvez déployer et configurer la passerelle REST pour la grille de données dans WebSphere Application Server ou dans un serveur profil Liberty.

Avant de commencer

Vérifiez qu'un serveur profil Liberty est créé. Pour plus d'informations, voir Installation de profil Liberty.

Pourquoi et quand exécuter cette tâche

La passerelle REST est un servlet qui est défini dans le fichier archive Web (WAR) `wxsRESTGateway.war`. Avec cette passerelle REST, vous utilisez un identificateur URI (Uniform Resource Identifier) pour accéder aux données dans la grille de données.

Procédure

1. Activez la fonction de passerelle REST en modifiant manuellement le fichier `server.xml` ou en utilisant Liberty Profile Developer Tools.
   - Activez la passerelle REST dans le fichier profil Liberty `server.xml`.

     ```
     <featureManager>
       <feature>eXtremeScale.rest-1.1</feature>
     </featureManager>
     ```

   - Activez la passerelle REST dans le fichier profil Liberty `server.xml` en utilisant Liberty Profile Developer Tools.
     - Démarrez IBM®WebSphere Application Server Version 8.6 Liberty Profile Developer Tools.
     - Dans l'onglet **Design**, sélectionnez **Feature Manager**. Cliquez sur **Add** dans la section Feature Manager Details. Sélectionnez la fonction **eXtremeScale.rest-1.1** et ajoutez-la.
     - Feature Manager étant sélectionné, cliquez sur **Add** dans la section Feature Manager Details. Sélectionnez la fonction **servlet-3.0** et ajoutez-la.
     - Enregistrez le fichier `server.xml`.
   - Activez la passerelle REST dans WebSphere Application Server.
     - Installez WebSphere eXtreme Scale avec WebSphere Application Server. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client avec WebSphere Application Server.
     - Déployez le fichier *was_install_root*`\optionalLibraries\ObjectGrid\restgateway\wxsRESTGateway.war` sur WebSphere Application Server.
2. Configurez la passerelle REST.
   a. Configurez la passerelle REST dans le fichier `server.xml`. Entrez la ligne de code suivante :

      ```
      <xsREST contextRoot=”myContextRoot” remoteDomain=”myDomain”/>
      ```

      Avertissement : Les attributs, `contextRoot` et `remoteDomain`, sont facultatifs. La racine de contenu par défaut est `resources`.

   b. Configurez un serveur eXtreme Scale.

   c. Configurez un serveur de conteneur.

      Les options suivantes sont disponibles pour configurer un service de conteneur :
      - Copiez un fichier valide `objectgrid.xml` (avec ou sans fichier `objectGridDeployment.xml` correspondant) dans le répertoire *rép_base_wlp*`/usr/servers/`*nom_serveur*`/grids`. Ce répertoire `grids` est surveillé par le produit lors de l'exécution. Les modifications des fichiers dans ce répertoire génèrent des événements dans l'environnement d'exécution profil Liberty. Par exemple, lorsqu'un nouveau fichier `objectgrid.xml` ou `objectGridDeployment.xml` ou ces deux nouveaux fichiers sont détectés, un serveur de conteneur est créé. Lorsque l'un de ces fichiers est supprimé, eXtreme Scale arrête ce serveur de conteneur. Lorsque les fichiers sont modifiés, eXtreme Scale arrête et redémarre le conteneur. Plusieurs conteneurs de fragment peuvent exister dans un même serveur eXtreme Scale, ce qui implique que des sous-répertoires existent dans le répertoire `grids`.
3. Démarrez le serveur profil Liberty pour exécuter la passerelle client REST.

Que faire ensuite

Lorsque la passerelle REST est activée, un utilisateur ayant accès au servlet peut accéder aux données dans une grille de données. Par conséquent, vous devez utiliser la sécurité d'application Web dans WebSphere Application Server pour contrôler l'autorisation. Pour plus d'informations sur la sécurisation des applications Web qui utilisent cette passerelle REST, voir Securing web applications dans le centre de documentation de WebSphere Application Server.

Le fichier `wxsRESTGateway.war`, qui contient le fichier `web.xml` pour la configuration de la sécurité, se trouve dans les emplacements suivants en fonction de votre installation :

- *`wlp_install_root`*`/wxs/web/rest`
- *`was_install_root`*`/optionalLibraries/ObjectGrid/restgateway`
- *`wxs_standalone_install_root`*`/ObjectGrid/restgateway`

Maintenant, vous pouvez utiliser le service de données Web dans le profil Liberty pour communiquer avec la grille de données via un identificateur URI. Pour plus d'informations, voir Développement d'applications de grille de données avec la passerelle REST.

Rubrique parent : Configuration des clients Java

.NET

Configuration de WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET à l'aide du fichier de propriétés du client, de la configuration XML côté serveur, ou en remplaçant à l'aide d'un programme certaines propriétés du serveur.

Pourquoi et quand exécuter cette tâche

Lorsque le client appelle la méthode Connect sur la grille de données, la configuration est effectuée à l'aide du fichier de propriétés du client spécifié. Si vous n'avez pas spécifié de fichier de propriétés, le fichier `Client.Net.properties` est utilisé.

Si des modifications sont apportées au fichier de propriétés du client après que le client a appelé la méthode Connect, vous devez redémarrer la connexion du client à la grille de données pour que ces modifications soient prises en compte.

Vous pouvez configurer la configuration dynamique de certaines propriétés dans le fichier `Client.Net.properties`.

La méthode Connect peut émettre des exceptions si le fichier de configuration du client contient des valeurs de propriété erronées, des propriétés mal placées ou d'autres erreurs. WebSphere eXtreme Scale Client for .NET contient également plusieurs valeurs de propriété de configuration XML côté serveur qui peuvent être remplacées par programme à l'aide de l'API Client for .NET.

.NET 2.5+ **Substitutions pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer WebSphere eXtreme Scale Client for .NET en remplaçant certaines propriétés du serveur. Ces propriétés sont définies dans le fichier de configuration `objectgrid.xml`.

.NET 2.5+ **Configuration de WebSphere eXtreme Scale Client for .NET à l'aide d'un programme**
Vous pouvez remplacer les paramètres côté client à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

.NET 2.5+ **Activation de la configuration dynamique de WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer WebSphere eXtreme Scale Client for .NET pour qu'il détecte dynamiquement les modifications apportées aux valeurs des propriétés du fichier de propriétés du client. Il n'est pas nécessaire de redémarrer la connexion à la grille de données pour que ces modifications prennent effet.

Rubrique parent : Configuration des clients

.NET

Substitutions pour WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET en remplaçant certaines propriétés du serveur. Ces propriétés sont définies dans le fichier de configuration `objectgrid.xml`.

Pour remplacer des paramètres sur un client, vous pouvez procéder à une configuration par programmation sur WebSphere eXtreme Scale Client for .NET. Pour plus d'informations sur les substitutions à l'aide d'un programme, voir Configuration de WebSphere eXtreme Scale Client for .NET à l'aide d'un programme.

Vous pouvez remplacer les attributs suivants sur un client :

Attributs BackingMap

- attribut timeToLive
- attribut LockTimeOut

Attributs ObjectGrid

- attribut txTimeout
- attribut txIsolation

Rubrique parent : .NET 2.5+ Configuration de WebSphere eXtreme Scale Client for .NET

.NET

Configuration de WebSphere eXtreme Scale Client for .NET à l'aide d'un programme

Vous pouvez remplacer les paramètres côté client à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur.

Pourquoi et quand exécuter cette tâche

L'exemple de code ci-dessous crée les mêmes substitutions que celles décrites dans Substitutions pour WebSphere eXtreme Scale Client for .NET.

Procédure

Le code suivant crée une instance ObjectGrid côté client :

```
IGridManager gm = GridManagerFactory.GetGridManager( );
ICatalogDomainInfo cdi =
gm.CatalogDomainManager.CreateCatalogDomainInfo("localhost:2809");
ctx = gm.Connect(cdi, "Sample.Client.Net.properties");
IGrid grid = gm.GetGrid( ctx, "Grid");

//Overriding grid’s txTimeout value
grid.TransactionTimeout = new TimeSpan(0, 0, 30);

IGridMapPessimisticTxObject, Object gridMap;
gridMap = grid.GetGridMapPessimisticTx<Object, Object>("Map1");

//Overriding timeToLive value
gridMap.TimeToLive = new TimeSpan(0, 0, 50);

//Overriding lockTimeout value
gridMap.LockTimeout = new TimeSpan(0, 0, 20);

//Overriding txTimeout value
gridMap.Transaction.TransactionTimeout = new TimeSpan(0, 0, 40);

//Overriding txIsolation value
gridMap.Transaction.TransactionIsolationLevel = TxnIsolationLevel.ReadUncommitted;

//Calling ResetToDefaults(), resets the value of timeToLive, lockTimeout,
//txTimeout, txIsolation back to values when .NET client initialized the grid.

gridMap.ResetToDefaults();
```

Avant de remplacer la valeur **timeToLive** à l'aide d'un programme, définissez la valeur **ttlEvictorType** sur LAST_ACCESS_TIME ou LAST_UPDATE_TIME dans le fichier `objectgrid.xml`. Si WebSphere eXtreme Scale Client for .NET tente de remplacer **timeToLive** à l'aide d'un programme sans définir la valeur **ttlEvictorType**, une exception est émise. Lorsque **txTimeout** a la valeur `TimeSpan.Zero`, le délai d'attente est illimité.

Rubrique parent : .NET 2.5+ Configuration de WebSphere eXtreme Scale Client for .NET

Information associée:

Propriété IGridTransaction.TransactionTimeout
Propriété IGridMapPessimisticAutoTx(Of TKey, TValue).LockTimeout
Propriété IGridMapPessimisticAutoTx(Of TKey, TValue).TimeToLive
Propriété IGridMapPessimisticTx(Of TKey, TValue).LockTimeout
Propriété IGridMapPessimisticTx(Of TKey, TValue).TimeToLive
Propriété IGrid.TransactionTimeout

.NET

Activation de la configuration dynamique de WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer WebSphere eXtreme Scale Client for .NET pour qu'il détecte dynamiquement les modifications apportées aux valeurs des propriétés du fichier de propriétés du client. Il n'est pas nécessaire de redémarrer la connexion à la grille de données pour que ces modifications prennent effet.

Pourquoi et quand exécuter cette tâche

Vous pouvez configurer dynamiquement la propriété requestRetryTimeout afin de spécifier la durée (en millisecondes) pendant laquelle le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Toutes les autres propriétés sont en lecture seule. Si vous modifiez la valeur d'une propriété en lecture seule, la modification n'est pas prise en compte et une erreur est consignée dans les fichiers journaux système.

Procédure

1. Dans votre fichier de propriétés du client, affectez la valeur `true` à la propriété enableDynamicConfiguration pour WebSphere eXtreme Scale Client for .NET.

   .NET 2.5+ **enableDynamicConfiguration**

   Lorsque cette propriété a la valeur `true`, les modifications apportées à la propriété requestRetryTimeout du fichier de propriétés du client sont détectées dynamiquement. La nouvelle valeur de cette propriété est immédiatement utilisée pour calculer le nouveau délai d'attente pour les nouvelles tentatives d'exécution de requête.

   Valeur par défaut : `false`

2. Mettez à jour la valeur de la propriété requestRetryTimeout dans votre fichier de propriétés du client.

   Java .NET **requestRetryTimeout**

   Indique pendant combien de temps (en millisecondes) le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Utilisez l'une des valeurs admises suivantes :
   - Une valeur égale à `0` indique que la requête doit échouer immédiatement et ignorer la logique interne régissant les nouvelles tentatives.
   - Une valeur égale à `-1` indique que le délai entre les tentatives d'exécution de la requête n'est pas défini, ce qui signifie que la durée pendant laquelle le système tente d'exécuter la requête dépend du délai d'expiration des transactions. (Valeur par défaut)
   - Une valeur supérieure à `0` indique la valeur du délai d'expiration de la requête en millisecondes. Les exceptions dont la création échoue sont renvoyées. Même lorsque des exceptions, telles que DuplicateException, sont réexécutées, elles sont également renvoyées quand elles échouent. Le délai de transaction reste utilisé comme délai d'attente maximal.

Résultats

Vous pouvez mettre à jour dynamiquement la valeur de la propriété requestRetryTimeout du client sans redémarrer la connexion à la grille de données.

Rubrique parent : .NET 2.5+ Configuration de WebSphere eXtreme Scale Client for .NET

Administration des grilles de données

Vous pouvez utiliser la console, la ligne de commande et l'interface de commande HTTP pour administrer vos grilles de données.

Demande, affichage et invalidation des données
Vous pouvez utiliser les interfaces de requête dans la console de surveillance et dans l'utilitaire **xscmd** pour extraire de petits ensembles de clés et de valeurs à partir d'une mappe et invalider des ensembles de données.

Administration avec l'utilitaire xscmd
Utilisez l'utilitaire **xscmd** pour effectuer des tâches d'administration dans l'environnement.

Administration à l'aide de l'interface de commande HTTP
A l'aide de l'interface de commande HTTP, vous pouvez effectuer des opérations sur votre dispositif, configurer les paramètres du dispositif et administrer des grilles de données, des collectivités et des zones.

Demande, affichage et invalidation des données

Vous pouvez utiliser les interfaces de requête dans la console de surveillance et dans l'utilitaire **xscmd** pour extraire de petits ensembles de clés et de valeurs à partir d'une mappe et invalider des ensembles de données.

Avant de commencer

- Si vous utilisez **xscmd** pour interroger afficher et invalider des données, configurez l'utilitaire **xscmd**. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser la console ou l'utilitaire **xscmd** pour interroger le contenu d'une grille de données. Vous pouvez interroger les données en exécutant une expression régulière sur la clé de données. Vous pouvez ensuite utiliser la même requête pour invalider les données. Pour des exemples d'expressions régulières voir Syntaxe d'expression régulière.

Procédure

- Demandez affichez ou invalidez des données avec la console.
  1. Accédez à la page de requête dans la console. Dans l'interface utilisateur, cliquez sur **Gestion de données** > **Contenu de la grille de données de requête**.
  2. Recherchez ou filtrez les données dans la mappe. Vous pouvez utiliser l'une des options suivantes pour rechercher ou filtrer les données :
     - Tapez une expression régulière dans la zone, puis cliquez sur le bouton **Rechercher** ( ). Une liste de clés correspondant à l'expression régulière s'affiche. La liste des données peut être un sous-ensemble de toutes les données correspondantes.
     - Pour filtrer les résultats en fonction d'un ensemble de partitions, cliquez sur le bouton **Filtrer** ( ). Vous pouvez ensuite taper une expression régulière et choisir une plage de partitions en fonction de laquelle vous voulez filtrer les résultats.
  3. Affichez les valeurs des clés affichées. Sélectionnez **Afficher les valeurs**. Les valeurs s'affichent dans le tableau. Si la valeur est trop longue à afficher, des points de suspension (. . .) tronquent la valeur. Cliquez sur la valeur pour afficher toute la zone. Les valeurs sont renvoyées sous forme de chaînes de texte. Certaines valeurs peuvent ne pas être converties en chaînes explicites et des valeurs hexadécimales s'affichent.

     Important : L'application peut stocker des valeurs d'objet pour lesquelles la classe Java™ n'est pas reconnue par le serveur. Si l'application utilise eXtreme Data Format (XDF), ces valeurs s'affichent. Si XDF n'est pas utilisé et que la classe Java n'est pas reconnue par le serveur, un message indique que la classe de l'objet n'est pas disponible pour le serveur.
  4. Invalidez les données. Lors de l'invalidation des données, celles-ci sont définitivement supprimées de la grille de données.

     Clés sélectionnées

     Vous pouvez sélectionner dans la table des clés à invalider. Vous pouvez alors cliquer individuellement sur les entrées ou cocher la case Tout sélectionner qui permet de sélectionner un maximum de 500 entrées dans la table. Une fois les entrées à supprimer sélectionnées, cliquez sur **Invalider** > **Clés sélectionnées**.

     Toutes les clés correspondant à la requête

     Vous pouvez également invalider toutes les données correspondant à votre expression régulière. Cette option permet de supprimer toutes les données de la grille de données qui correspondent à l'expression régulière, et pas uniquement le nombre maximal de 500 entrées affichées dans la console. Pour invalider des entrées comportant l'expression régulière, cliquez sur **Invalider** > **Toutes les clés correspondant à la requête**.
  5. Supprimez tout le contenu de la mappe. Cliquez sur **Effacer la mappe**. Vous devez confirmer la suppression de toutes les entrées dans la mappe sélectionnée.
- Demandez affichez ou invalidez des données avec l'utilitaire **xscmd**.

  Interrogation de données :

  ```
  xscmd.sh –c findbykey –g <grille_données> -m <mappe>
  -fs <chaîne_recherche> [-fp <id_partition>]
  ```

  Vous devez inclure la grille de données, la mappe et l'expression régulière pour la valeur de la chaîne

de recherche. Vous pouvez également appliquer un filtrage par ID partition. Le résultat renvoie un sous-ensemble de la totalité de la requête.

Invalidation de données :

Incluez l'argument **-inv** dans la commande pour invalider les données sélectionnées par la requête.

```
xscmd –c findbykey –g <grille_données> -m <mappe>
-fs <chaîne_recherche> [-fp <id_partition>] -inv
```

Vous devez inclure la grille de données, la mappe et l'expression régulière pour la valeur de la chaîne de recherche. Vous pouvez également appliquer un filtre par ID partition. Lors de l'exécution de l'invalidation, toutes les valeurs correspondantes sont invalidées, et pas seulement le petit ensemble renvoyé par la requête.

Affichez les valeurs des données demandées :

Incluez l'argument **-rv** dans la commande pour afficher les valeurs des données sélectionnées par la requête.

```
xscmd.sh –c findbykey –g <grille_données> -m <mappe>
-fs <find_string> -rv
```

Vous devez inclure la grille de données, la mappe et l'expression régulière pour la valeur de la chaîne de recherche. Vous pouvez également appliquer un filtre par ID partition. Le résultat renvoie un sous-ensemble de la totalité de la requête et inclut les valeurs de chaque clé.

UNIX Linux **Important :** Si votre expression régulière commence par les caractères `.*`, il se peut que ces caractères ne soient pas traités correctement lors de l'exécution de la commande. Pour résoudre ce problème, mettez en forme votre expression régulière d'une des manières suivantes :

- Placez votre expression régulière entre des apostrophes : `-fs '.*'`
- Utilisez une barre oblique inversée comme caractère d'échappement pour l'astérisque : `-fs .\*`

Exemple :

Dans l'exemple ci-dessous, toutes les entrées de la grille de données `Grid` et de la mappe `Map1` sont recherchées.

```
xscmd -c findbykey -g Grid -m Map1 -fs ".*"
```

La commande renvoie les résultats suivants :

```
3 matching keys were found.

Partition Key
--------- ---
2         keyghi
4         keydef
6         keyabc
```

Rubrique parent : Administration des grilles de données

Tâches associées:
Administration avec l'utilitaire xscmd

Référence associée:
Référence à l'utilitaire xscmd

Administration avec l'utilitaire `xscmd`

Utilisez l'utilitaire **`xscmd`** pour effectuer des tâches d'administration dans l'environnement.

Avant de commencer

- Le dispositif doit être démarré.
- Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client : Vous devez connaître l'adresse IP et le numéro de port d'un serveur de catalogue actif. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Membres**. Sélectionnez un membre de collectivité. L'adresse IP et le numéro de port du serveur de catalogue s'affichent.
- Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client : Vérifiez que la variable d'environnement *`JAVA_HOME`* est définie pour utiliser l'environnement d'exécution installé avec le produit. Si vous utilisez la version d'évaluation du produit, vous devez définir la variable d'environnement *`JAVA_HOME`*.

Pourquoi et quand exécuter cette tâche

2.5+ Vous pouvez exécuter l'utilitaire **`xscmd`** à partir d'une installation client ou de l'interface de ligne de commande du dispositif. Lorsque vous l'exécutez à partir de l'interface de ligne de commande, il se connecte automatiquement à un serveur de catalogue de la collectivité. Dans ce cas, vous n'avez pas besoin de définir les variables d'environnement et d'importer le fichier de clés certifiées du dispositif. Vous ne pouvez vous connecter qu'à la collectivité locale. Si vous voulez vous connecter à des collectivités éloignées, vous devez opérer à partir d'une installation client ou utiliser l'interface de ligne de commande de l'un des dispositifs de la collectivité éloignée concernée.

Procédure

1. Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client : Téléchargez sur le client le fichier de clés certifiées actif du dispositif. Dans l'interface utilisateur du dispositif, cliquez sur **Collectivité** > **Paramètres** > **TLS (Transport Layer Security)** > **Télécharger le fichier de clés certifiées actif**. Le fichier de clés certifiées par défaut est le fichier `xsatruststore.jks`. Le mot de passe par défaut est `xc10pass`.
2. Facultatif : Si l'authentification de client est activée : Sur l'installation client, ouvrez une fenêtre de ligne de commande. Sur la ligne de commande, définissez les variables d'environnement appropriées.
3. Connectez l'utilitaire **`xscmd`** au dispositif.
   - Si vous exécutez l'utilitaire **`xscmd`** à partir d'une installation client :

     Dans le répertoire `bin` de l'installation client, exécutez la commande suivante :

     ```
     xscmd.bat|sh -ts xsatruststore.jks -tst jks -tsp xc10pass -user xcadmin -pwd
     xcadmin
     -cep myxc10.mycompany.com  -prot TLS -cxpv IBMJSSE2 -tt TCP/IP [paramètre
     supplémentaires]
     ```

   - 2.5+ Si vous exécutez l'utilitaire **`xscmd`** à partir de l'interface de ligne de commande du dispositif :
     a. Connectez-vous à l'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.
     b. Exécutez l'utilitaire **`xscmd`**:

        La syntaxe générale de cette commande est la suivante :

        ```
        Console> xscmd -c <nom_commande> -opt1 [arg1] -opt2 [arg2] -opt3
        ```

        La commande suivante affiche l'aide du dispositif :

        ```
        Console> xscmd -h
        ```

4. Affichez l'aide des différentes options **`xscmd`**. Si vous exécutez l'utilitaire **`xscmd`** à partir de l'interface de ligne de commande du dispositif, l'extension `.bat|.sh` n'est pas nécessaire.
   - Pour afficher l'aide générale, exécutez la commande suivante :
     - UNIX `./xscmd.sh -h`
     - Windows `xscmd.bat -h`
   - Pour afficher la liste de toutes les commandes, exécutez la commande suivante :
     - UNIX `./xscmd.sh -lc`

        - Windows `xscmd.bat -lc`
    - Pour afficher l'aide d'une commande, exécutez la commande suivante :
        - UNIX `./xscmd.sh -h nom_commande`
        - Windows `xscmd.bat -h nom_commande`
    - Pour afficher une liste des groupes de commandes, exécutez la commande suivante :
        - UNIX `./xscmd.sh -lcg`
        - Windows `xscmd.bat -lcg`
    - Pour afficher la liste des commandes d'un groupe de commandes, exécutez la commande suivante :
        - UNIX `./xscmd.sh -lc nom_groupe_commandes`
        - Windows `xscmd.bat -lc nom_groupe_commandes`
5. Exécutez les commandes de connexion à de serveurs de catalogue spécifiques. Vous devez fournir une ou plusieurs combinaisons de port et d'adresse IP de serveur de catalogue pour extraire des informations relatives aux grilles de données exécutées sur le dispositif.Lorsque vous utilisez l'interface de ligne de commande du dispositif, vous ne pouvez vous connecter qu'à la collectivité locale. Si vous voulez vous connecter à des collectivités éloignées, vous devez opérer à partir d'une installation client ou utiliser l'interface de ligne de commande de l'un des dispositifs de la collectivité éloignée concernée.
    - Fournissez la liste des serveurs de catalogue auxquels vous voulez vous connecter :
        - UNIX `./xscmd.sh -c <nom_commande> -cep nom_hôte:port(,nom_hôte:port)`
        - Windows `xscmd.bat -c <nom_commande> -cep nom_hôte:port(,nom_hôte:port)`

      Dans les commandes précédentes, *nom_commande* désigne le nom de la commande que vous exécutez. La valeur *nom_hôte:port* représente le nom d'hôte du serveur de catalogue et le port d'écoute.

ATTENTION :

N'utilisez pas les commandes suivantes dans un environnement WebSphere DataPower XC10 Appliance :

- **-c releaseShard**
- **-c reserveShard**
- **-c swapShardWithPrimary**
- **-c suspendBalancing**
- **-c resumeBalancing**
- **-c teardown**
- **-c triggerPlacement**
- **-c enableForPlacement**

6. Facultatif : Définissez une valeur de délai d'attente lorsque vous exécutez vos commandes. Vous pouvez utiliser l'option **-to** ou **--timeout** comme paramètre global dans n'importe quelle commande. Cette valeur définit le nombre de secondes avant l'expiration du délai d'attente lorsque vous vous connectez à des serveurs de catalogue dans la commande. Si vous vous connectez à un serveur de catalogue qui risque d'être indisponible à la suite de l'expiration d'un délai d'attente du système d'exploitation ou du réseau, cette option peut être utile pour réduire l'attente.

   Le délai d'attente par défaut est de 30 secondes.

Configuration des profils de sécurité pour l'utilitaire xscmd
En créant un profil de sécurité, vous pouvez utiliser les paramètres de sécurité enregistrés pour utiliser l'utilitaire **xscmd** avec des environnements sécurisés.

Rubrique parent : Administration des grilles de données

Tâches associées:
Demande, affichage et invalidation des données

Référence associée:
Référence à l'utilitaire xscmd

Configuration des profils de sécurité pour l'utilitaire `xscmd`

En créant un profil de sécurité, vous pouvez utiliser les paramètres de sécurité enregistrés pour utiliser l'utilitaire **`xscmd`** avec des environnements sécurisés.

Avant de commencer

Pour plus d'informations sur la configuration de l'utilitaire **`xscmd`**, voir Administration avec l'utilitaire xscmd.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser le paramètre **`-ssp`** *`nom_profil`* ou **`--saveSecProfile`** *`nom_profil`* avec le reste de la commande **`xscmd`** pour enregistrer un profil de sécurité. Le profil peut contenir des paramètres pour les noms d'utilisateur et les mots des générateurs de données d'identification, des fichiers de clés, des fichiers de clés certifiées et des types de transport.

Le groupe de commandes **`ProfileManagement`** dans l'utilitaire **`xscmd`** contient des commandes de gestion de vos profils de sécurité.

Procédure

- Enregistrez un profil de sécurité.

  Pour enregistrer un profil de sécurité, utilisez le paramètre **`-ssp`** *`nom_profil`* ou **`--saveSecProfile`** *`nom_profil`* avec le reste de la commande. L'ajout de ce paramètre à la commande enregistre les paramètres suivants :

  ```
  -al,--alias <alias>
  -arc,--authRetryCount <integer>
  -ca,--credAuth <support>
  -cgc,--credGenClass <className>
  -cgp,--credGenProps <property>
  -cxpv,--contextProvider <provider>
  -ks,--keyStore <filePath>
  -ksp,--keyStorePassword <password>
  -kst,--keyStoreType <type>
  -prot,--protocol <protocol>
  -pwd,--password <password>
  -ts,--trustStore <filePath>
  -tsp,--trustStorePassword <password>
  -tst,--trustStoreType <type>
  -tt,--transportType <type>
  -user,--username <username>
  ```

  Les profils de sécurité sont enregistrés dans le répertoire *`rép_base_utilisateur`*`\.scmd\profiles\security\<`*`nom_profil`*`>.properties`.

  Important : N'incluez pas l'extension de nom de fichier `.properties` dans le paramètre *`nom_profil`*. Cette extension est automatiquement ajoutée au nom de fichier.

- Utilisez un profil de sécurité enregistré.

  Pour utiliser un profil de sécurité enregistré, ajoutez le paramètre **`-sp`** *`nom_profil`* ou **`--securityProfile`** *`nom_profil`* à la commande que vous exécutez.

  Exemple de commande : `xscmd -c listHosts -cep myhost.mycompany.com -sp myprofile`

- Listez les commandes dans le groupe de commandes **`ProfileManagement`**.

  Exécutez la commande **`xscmd -lc ProfileManagement`**.

- Listez les profils de sécurité existants.

  Exécutez la commande **`xscmd -c listProfiles -v`**.

- Affichez les paramètres enregistrés dans un profil de sécurité.

  Exécutez la commande **`xscmd -c showProfile -pn`** *`nom_profil`*.

- Supprimez un profil de sécurité existant.

  Exécutez la commande **`xscmd -c RemoveProfile -pn`** *`nom_profil`*.

Rubrique parent : Administration avec l'utilitaire xscmd

Référence associée:
Référence à l'utilitaire xscmd

Administration à l'aide de l'interface de commande HTTP

A l'aide de l'interface de commande HTTP, vous pouvez effectuer des opérations sur votre dispositif, configurer les paramètres du dispositif et administrer des grilles de données, des collectivités et des zones.

Pourquoi et quand exécuter cette tâche

L'interface de commande HTTP permet d'effectuer des opérations avec les instructions JSON HTTP POST. Vous pouvez regrouper ces instructions dans des scripts pour automatiser les tâches de configuration et d'administration.

Procédure

1. Visualisez les commandes disponibles pour l'interface de commande HTTP dans l'interface utilisateur.
   Pour visualiser une table de toutes les commandes disponibles dans l'interface utilisateur, cliquez sur **(Aide)** > **Interface de commande HTTP - Aide**. Cliquez sur un nom de commande pour afficher la **Syntaxe détaillée** et un **Exemple de soumission JSON**.

   Pour afficher la liste de toutes les commandes dans le centre de documentation, voir Guide de référence de l'interface de commande HTTP.
2. Créez une instruction JSON pour l'opération que vous souhaitez effectuer. Cette instruction doit contenir :
   - La commande que vous souhaitez exécuter.
   - Les paramètres appropriés pour la commande.

   Utilisez la commande suivante dans l'outil **cURL** :

   ```
   curl -v -k -u UTIL_ADMIN_XC:MDP_ADMIN_XC -H "Content-Type: application/json" --data-binary 'INSTRUCTION_JSON_INTERFACE_HTTP'
   https://NOM_HOTE_XC10/resources/appTaskInterface
   ```

   Définissez les variables suivantes :

   UTIL_ADMIN_XC:MDP_ADMIN_XC

   Indique le nom d'utilisateur et le mot de passe de l'administrateur WebSphere DataPower XC10 Appliance.

   INSTRUCTION_JSON_INTERFACE_HTTP

   Indique l'une des instructions JSON possibles soumises à l'interface de commande HTTP. Vous pouvez copier l'instruction **Exemple de soumission JSON** à partir de **Interface de commande HTTP - Aide** pour la commande que vous voulez exécuter. La commande doit être placées entre apostrophes (').

   NOM_HOTE_XC10

   Indique le nom d'hôte complet ou l'adresse IP de WebSphere DataPower XC10 Appliance.

   Par exemple, vous pouvez copier l'exemple de soumission JSON de la commande ViewAllUsers à partir de l'aide de l'interface de commande HTTP. Puis vous pouvez exécuter la commande suivante :

   ```
   curl -v -k -u xcadmin:xcadmin -H "Content-Type: application/json" --data-binary
   '{"task":{"stopOnTaskFailure":"true","command":"ViewAllUsers"}}'
   https://myXC10.mycompany.com/resources/appTaskInterface
   ```
3. Exécutez la commande et consultez sa sortie.

Rubrique parent : Administration des grilles de données

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Information associée:
Outils cURL et libcurl
Guide de référence de l'interface de commande HTTP

Développement d'applications pour accéder à des grilles de données simples

Vous pouvez utiliser deux options différentes pour créer des mappes et lire, écrire et mettre à jour des grilles de données simples. Vous pouvez écrire une application client Java qui utilise l'API ObjectMap. Vous pouvez aussi utiliser la passerelle REST pour développer une application non-Java pour accéder à la grille de données simple.

Avant de commencer

Vous devez disposer d'une grille de données simple existante. Pour plus d'informations sur la création d'une grille de données simple, voir Création de grilles de données simples.

Si vous utilisez une mémoire cache dynamique ou une grille de données de session, vous pouvez utiliser vos applications existantes sans modification.

Développement d'applications de grilles de données avec des API Java
Vous pouvez vous connecter au serveur de catalogue, obtenir une instance ObjectGrid et utiliser l'API ObjectMap.

Développement d'applications de grille de données avec la passerelle REST
Vous pouvez utiliser la passerelle REST (Representational State Transfer) pour accéder à des grilles de données simples qui sont hébergées par une collectivité. Cette passerelle REST est pratique lorsque vous devez accéder à une grille de données à partir d'environnements non Java.

.NET 2.5+ **Développement d'applications de grille de données avec les API .NET**
Vous pouvez développer des applications Microsoft .NET qui utilisent la même grille de données que vos applications Java™.

Développement d'applications de grilles de données avec des API Java

Vous pouvez vous connecter au serveur de catalogue, obtenir une instance ObjectGrid et utiliser l'API ObjectMap.

Avant de commencer

Vous devez créer une grille de données simple dans l'interface utilisateur. Pour plus d'informations, voir Création de grilles de données simples.

Visite virtuelle d'application de grille de données simple

1. **Connectez-vous au service de catalogue via une instance ClientClusterContext.**

   Le serveur de catalogue à spécifier est affiché dans la page d'interface utilisateur de la grille de données simple que vous avez créée. Cliquez sur **Grille de données** > **Grille de données simple** > ***my_simple_data_grid*** et utilisez les valeurs dans la zone **Services de catalogue**.

   Pour établir la connexion au serveur de catalogues, utilisez la méthode connect de l'API ObjectGridManager. La méthode connect utilisée requiert seulement un noeud final de serveur de catalogue au format *nom_hôte*:*port*. Vous pouvez indiquer plusieurs noeuds finaux de serveur de catalogue en séparant les valeurs *hostname*:*port* par une virgule. Le fragment de code suivant montre comment se connecter à un serveur de catalogue et obtenir une instance ClientClusterContext :

   ```
   ClientClusterContext ccc =
   ObjectGridManagerFactory.getObjectGridManager().connect("myXC10.myhost.com:2809",
   null, null);
   ```

   La méthode connect tente de se connecter à chaque dispositif de la liste jusqu'à ce qu'elle puisse établir une connexion. Un basculement automatique est effectué si l'un des autres dispositifs ne répond pas.

   Si les connexions aux serveurs de catalogue aboutissent, la méthode connect retourne une instance ClientClusterContext. L'instance ClientClusterContext est requise pour obtenir l'ObjectGrid à partir de l'API ObjectGridManager.
2. **Obtenez une instance ObjectGrid.**

   Pour obtenir une instance ObjectGrid, utilisez la méthode getObjectGrid de l'API ObjectGridManager. La méthode getObjectGrid requiert l'instance ClientClusterContext et le nom de l'instance de grille de données. L'instance ClientClusterContext est obtenue pendant la connexion au serveur de catalogue. Le nom de l'instance de grille de données est le nom de la grille de données simple que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir la grille de données en appelant la méthode getObjectGrid de l'API ObjectGridManager.

   ```
   ObjectGrid grid = ObjectGridManagerFactory.getObjectGridManager().getObjectGrid(ccc,
   “my_simple_data_grid”);
   ```

3. **Définissez les droit d'accès de sécurité nécessaires.**

   Créez une configuration de sécurité du client avec un nom d'utilisateur et un mot de passe que vous fournissez à l'application. Le nom d'utilisateur et le mot de passe que vous utilisez doivent avoir l'autorisation d'accéder à la grille de données sur le dispositif. Pour plus d'informations sur la création d'un utilisateur autorisé, voir Gestion des utilisateurs et des groupes.

   ```
                  // Creates a ClientSecurityConfiguration object using the specified
   file
                  ClientSecurityConfiguration clientSC =
   ClientSecurityConfigurationFactory.getClientSecurityConfiguration();
                  clientSC.setSecurityEnabled(true);
                  // Creates a CredentialGenerator using the passed-in user and
   password.
                  CredentialGenerator credGen = new
   UserPasswordCredentialGenerator(username,password);
                  clientSC.setCredentialGenerator(credGen);
                  return clientSC;
   ```

4. **Obtenez une instance Session.**

   Vous pouvez obtenir une session de l'instance ObjectGrid obtenue. Une instance Session est

indispensable pour obtenir l'instance ObjectMap et pour effectuer une démarcation de transaction. Le fragment de code suivant montre comment obtenir une instance Session en appelant la méthode getSession de l'API ObjectGrid.

```
Session sess = grid.getSession();
```

5. **Obtenez une instance ObjectMap.**

   Après avoir obtenu une instance Session, vous pouvez obtenir une instance ObjectMap à partir d'une instance Session en appelant la méthode getMap de l'API Session. Le nom d'instance de mappe que vous envoyez à la méthode getMap porte le nom de la grille de données que vous avez créée dans l'interface utilisateur. Le fragment de code suivant montre comment obtenir ObjectMap en appelant la méthode getMap de l'API Session.

   ```
   ObjectMap map1 = sess.getMap("my_simple_data_grid");
   ```

   L'exemple précédent utilise l'instance de mappe par défaut qui est nommé d'après la grille de données. Vous pouvez également indiquer un nouveau nom de mappe, comme dans les exemples suivants :

   ```
   ObjectMap map2 = sess.getMap("my_simple_data_grid.CT.P");
   ObjectMap map3 = sess.getMap("my_new_map.NONE");
   ```

   La mappe `my_simple_data_grid.CT.P` est une mappe qui utilise l'expulsion en fonction de l'heure de création et le verrouillage pessimiste. La mappe `my_new_map.NONE` ne dispose pas de paramètres d'expulsion ou de verrouillage. Pour plus d'informations, voir Options de configuration de mappe dynamique.
6. **Utilisez les méthodes ObjectMap.**

   Une fois une instance ObjectMap obtenue, vous pouvez utiliser l'API ObjectMap. N'oubliez pas que l'interface ObjectMap est une mappe transactionnelle et qu'elle requiert une démarcation de transaction à l'aide des méthodes begin et commit de l'API Session. Faute de démarcation de transaction explicite, les opérations ObjectMap s'exécutent avec des transactions de validation automatique.

   Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une transaction de validation automatique.

   ```
   map1.insert(key1, value1);
   ```

   La clé que vous utilisez peut avoir n'importe quel type Java existant, tel que java.lang.String ou Entier. Les valeurs peuvent correspondre à n'importe quel type d'objet sérialisable.

   Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une démarcation de transaction explicite.

   ```
   sess.begin();
   map1.insert(key1, value1);
   sess.commit();
   ```

Accès à la documentation des API Java

Vous pouvez accéder à la documentation des API Java™ pour WebSphere DataPower XC10 Appliance en téléchargeant un fichier archive zip, en intégrant la documentation des API Dans l'environnement de développement ou en l'affichant dans le centre de documentation.

Java Exemple : application grille de données simple

Cet exemple utilise l'API ObjectMap pour effectuer des opérations de création, de récupération, de mise à jour et de suppression simples sur la grille de données.

Création de mappes dynamiques avec les API Java

Vous pouvez créer des mappes dynamiques avec des API Java une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

2.5+ Programmation de transactions dans des applications Java

Lorsque vous écrivez une application Java qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

Plug-in d'indexation des données

Selon le type d'index que vous voulez générer, WebSphere eXtreme Scale Client fournit des plug-in intégrés que vous pouvez ajouter à la mappe de sauvegarde pour générer un index.

2.5+ **Notification aux clients des mises à jour de mappe en utilisant l'interrogation continue**
Vous pouvez être notifié dans votre machine JVM (Java virtual machine) client de l'insertion ou de la mise à jour d'objets ou d'entrées dans la grille de données.

Rubrique parent : Développement d'applications pour accéder à des grilles de données simples

Tâches associées:
Création de grilles de données simples

Référence associée:
Exemple : application grille de données simple
Fichier de propriétés du client

Information associée:
Spécification de l'API client

Java

Accès à la documentation des API Java

Vous pouvez accéder à la documentation des API Java™ pour WebSphere DataPower XC10 Appliance en téléchargeant un fichier archive zip, en intégrant la documentation des API Dans l'environnement de développement ou en l'affichant dans le centre de documentation.

Pourquoi et quand exécuter cette tâche

Vous pouvez accéder à la documentation des API Java dans l'un des emplacements suivants :

Centre de documentation

L'utilisation de la documentation d'API du centre de documentation sert à effectuer des recherches conjointement avec le reste des informations de produit WebSphere DataPower XC10 Appliance.

Archive de fichier zip

Vous pouvez télécharger ce fichier pour chaque édition. Vous pouvez ensuite comparer les outils pour déterminer quelles API ont changé d'une édition à l'autre. Vous pouvez également lier directement le fichier compressé dans vos projets Eclipse lors de la compilation au niveau du fichier objectgrid.jar. Ce lien permet d'intégrer la documentation d'API à l'environnement de développement intégré (IDE).

Format en ligne

Le format en ligne est un exemplaire publié de la documentation des API sur le site Web d'IBM®. Vous pouvez vous connecter directement à cette URL dans Eclipse. Le lien de la version en cours est toujours mis à niveau vers la dernière version, de sorte que vous pouvez automatiquement visualiser les corrections et les modifications de la documentation.

Procédure

- Consultez la documentation d'API dans le centre de documentation. Pour plus d'informations, voir Documentation d'API.
- Téléchargez une archive zip de la documentation d'API.

  Pour télécharger la documentation d'API afin de la consulter hors ligne, vous pouvez télécharger un fichier zip correspondant à l'édition appropriée, à partir de la page suivante : WebSphere DataPower XC10 Appliancewiki : Documentation d'API.
- Consultez le format en ligne de la documentation d'API. Vous pouvez associer un signet à un lien qui est est toujours mis à niveau vers la dernière version, ou vous pouvez vous connecter à une version spécifique. Pour obtenir une liste de liens, voir WebSphere DataPower XC10 Appliance wiki : Documentation d'API.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Java

Exemple : application grille de données simple

Cet exemple utilise l'API ObjectMap pour effectuer des opérations de création, de récupération, de mise à jour et de suppression simples sur la grille de données.

Exemple d'application

SimpleGrid.java

```
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.HashSet;
import java.util.BitSet;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.PrintWriter;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.Serializable;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import com.ibm.websphere.objectgrid.ClientClusterContext;
import com.ibm.websphere.objectgrid.ConnectException;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.ObjectGridException;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridRuntimeException;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.config.BackingMapConfiguration;
import com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory;
import com.ibm.websphere.objectgrid.config.ObjectGridConfiguration;
import com.ibm.websphere.objectgrid.plugins.TransactionCallbackException;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration;
import com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory;
import com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator;
import
com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator;
import com.ibm.websphere.objectgrid.server.CatalogServerProperties;

public class SimpleGrid {

        static String gridName = "test";
        static String mapName = gridName;
        static String username="xcadmin";
        static String password="xcadmin";
        static String hostName = "localhost";
        static ObjectGrid clientGrid=null;
        static ConcurrentLinkedQueue<Session> sessions = new
ConcurrentLinkedQueue<Session>();

        static synchronized public ObjectGrid getObjectGrid() {
                if (clientGrid == null) {
                        ClientClusterContext ccc = null;
                        try {
                           new
```

```
java.io.File(System.getProperty("java.io.tmpdir")).mkdirs();
                        } catch (Throwable t) {
                                t.printStackTrace();
                        }
                        ObjectGridManager ogm =
ObjectGridManagerFactory.getObjectGridManager();
                        ClientSecurityConfiguration clientSC = getAdminClientConfig();
                        List<ObjectGridConfiguration> ogConfigs=new
ArrayList<ObjectGridConfiguration>();
                        ObjectGridConfiguration lclGridConfig =
ObjectGridConfigFactory.createObjectGridConfiguration(gridName);
                        BackingMapConfiguration bmc =
ObjectGridConfigFactory.createBackingMapConfiguration(mapName);
                        bmc.setNumberOfBuckets(0);
                        lclGridConfig.addBackingMapConfiguration(bmc);
                        ogConfigs.add(lclGridConfig);
                        try {
                                ccc = ogm.connect(hostName+":2809", clientSC, null);
                        } catch (Throwable e) {
                                e.printStackTrace();
                        }
                        if (ccc != null) {
                           HashMap<String,List<ObjectGridConfiguration>> overrideMap = new
HashMap<String,List<ObjectGridConfiguration>>();
                           overrideMap.put(ccc.getClusterName(),ogConfigs);
                           ogm.setOverrideObjectGridConfigurations(overrideMap);
                           try {
                                clientGrid = ogm.getObjectGrid(ccc, gridName);
                           } catch (ObjectGridRuntimeException ogre) {
                                ogre.printStackTrace();
                           }
                        }
                }
                return clientGrid;
        }

        static public Session getSession() throws TransactionCallbackException,
                        ObjectGridException {
                Session session = sessions.poll();
                if (session == null && getObjectGrid()!=null) {
                        session = getObjectGrid().getSession();
                }
                if (session == null)
                        throw new IllegalStateException("unable to initialize connection
to objectgrid");
                return session;
        }

        static public void putSession(Session session) {
                if (session.isTransactionActive()) {
                        try {
                                session.rollback();
                        } catch (Exception e) {
                        }
                sessions.add(session);
        }

        public static ClientSecurityConfiguration getAdminClientConfig() {
                // Creates a ClientSecurityConfiguration object using the specified file
                ClientSecurityConfiguration clientSC =
ClientSecurityConfigurationFactory.getClientSecurityConfiguration();
                clientSC.setSecurityEnabled(true);
                // Creates a CredentialGenerator using the passed-in user and password.
                CredentialGenerator credGen = new
UserPasswordCredentialGenerator(username,password);
                clientSC.setCredentialGenerator(credGen);
                return clientSC;
```

```
        }

    public static void main(String args[]) throws Exception {
    for (int i=0;i<args.length;i++) {
         if (args[i].startsWith("-username:")) {
            username = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-password:")) {
            password = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-gridname:")) {
            gridName = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-mapname:")) {
            mapName = args[i].substring(args[i].indexOf(":") + 1);
         } else if (args[i].startsWith("-hostname:")) {
            hostName = args[i].substring(args[i].indexOf(":") + 1);
         } else {
            System.out.println("usage: SimpleGrid [optional args]");
            System.out.println("       -username:<nomutilisateur>");
            System.out.println("       -password:<motdepasse>");
            System.out.println("       -gridname:<nomgrille>");
            System.out.println("       -mapname:<nommappe>");
            System.out.println("       -hostname:<nomhôte>");
            System.exit(1);
         }
    }
    System.out.println("-----------------------------------------------------------------");
    System.out.println("Test de grille simple");
    System.out.println("-----------------------------------------------------------------");
    System.out.println("username     : " + username);
    System.out.println("password     : " + password);
    System.out.println("gridname     : " + gridName);
    System.out.println("mapname      : " + mapName);
    System.out.println("hostname     : " + hostName);
    System.out.println("-----------------------------------------------------------------");

    if (getObjectGrid() == null) {
        System.out.println("ERREUR : impossible de se connecter à objectgrid à "       +
hostName);
        System.exit(1);
    }
             Session session = getSession();
             ObjectMap map=session.getMap(mapName);
             session.begin();
             Object data = map.get("TestKey");
             if ( data != null )
                 map.remove("TestKey");
             map.insert("TestKey", "TestValue");
             session.commit();
             putSession(session);
     }
}
```

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Concepts associés:
Développement d'applications de grilles de données avec des API Java

Tâches associées:
Création de grilles de données simples

Information associée:
Spécification de l'API client

Java

Création de mappes dynamiques avec les API Java

Vous pouvez créer des mappes dynamiques avec des API Java une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Avant de commencer

Choisissez les options de configuration que vous souhaitez utiliser dans votre mappe dynamique. Pour plus d'informations, voir Options de configuration de mappe dynamique.

Procédure

Appelez la méthode Session.getMap(String).

Si vous transmettez une chaîne qui correspond à l'expression régulière d'un des modèles prédéfinis, la mappe appropriée est créée.

```
ObjectMap map2 = sess.getMap("my_simple_data_grid.CT.P");
ObjectMap map3 = sess.getMap("my_new_map.NONE");
```

La mappe my_simple_data_grid.CT.P est une mappe qui utilise l'expulsion en fonction de l'heure de création, le verrouillage pessimiste et l'invalidation du cache local. La mappe my_new_map.NONE ne dispose pas de paramètres d'expulsion ou d'invalidation du cache local.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données
Développement d'applications de grilles de données avec des API Java

Référence associée:
Options de configuration de mappe dynamique
Exemple de passerelle REST : Création de mappes dynamiques

Programmation de transactions dans des applications Java

Lorsque vous écrivez une application Java™ qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

Java 2.5+ **Interaction avec les données dans une transaction pour les applications Java**
Utilisez des sessions pour interagir avec les données à l'aide des opérations insert et update.

2.5+ **Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction**
Si vous répartissez les données dans plusieurs partitions dans la grille de données, vous pouvez lire et mettre à jour plusieurs partitions dans une seule transaction. Ce type de transaction, appelée transaction multipartition, utilise le protocole de validation en deux phases pour coordonner et restaurer la transaction en cas d'échec.

2.5+ **Utilisation du verrouillage**
Les verrous comportent des cycles de vie et leurs différents types sont compatibles entre eux selon plusieurs critères. Les verrous doivent être traités dans un ordre approprié pour éviter les situations d'interblocage.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Java

Interaction avec les données dans une transaction pour les applications Java

Utilisez des sessions pour interagir avec les données à l'aide des opérations insert et update.

Pourquoi et quand exécuter cette tâche

L'interface ObjectMap offre les opérations put, get et remove habituelles. Nous vous recommandons toutefois d'utiliser des noms d'opérations plus précis tels que get, getForUpdate, insert, update et remove. Ces méthodes permettent d'exécuter des opérations de façon plus précise que les API Map habituelles.

Java **Remarque :** Les méthodes upsert et upsertAll remplacent les méthodes ObjectMap put et putAll. Utilisez la méthode upsert pour indiquer à la mappe de sauvegarde qu'une entrée de la grille de données doit placer la clé et la valeur dans la grille. La mappe de sauvegarde exécute une insertion ou une mise à jour pour placer la valeur dans la grille .

Vous pouvez également utiliser la prise en charge de l'indexation, qui est flexible.

Procédure

- Insertion de données.

  Dès que vous avez obtenu une session, vous pouvez utiliser le fragment de code suivant pour insérer des données à l'aide de l'API Map :

  ```
  Session session = ...;
  ObjectMap personMap = session.getMap("PERSON");
  session.begin();
  Person p = new Person();
  p.name = "John Doe";
  personMap.insert(p.name, p);
  session.commit();
  ```

- Mise à jour de données.

  Vous pouvez utiliser le fragment de code suivant pour mettre à jour des données à l'aide de l'API Map.

  ```
  session.begin();
  Person p = (Person)personMap.getForUpdate("John Doe");
  p.name = "John Doe";
  p.age = 30;
  personMap.update(p.name, p);
  session.commit();
  ```

  L'application utilise normalement la méthode getForUpdate plutôt que la simple méthode get pour verrouiller un enregistrement. La méthode de mise à jour doit être appelée pour fournir la valeur mise à jour à la mappe. Si tel n'est pas le cas, la mappe n'est pas modifiée.

- 2.5+ Insertion de données avec le protocole de validation en deux phases en appelant la méthode `session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE); session.begin();`. Le fragment de code suivant montre comment créer, extraire, mettre à jour et supprimer des données dans une grille avec un protocole de validation en deux phases .

  ```
  Session session = og.getSession();
  Objectmap map1 = session.getMap("Map1");
  Objectmap map2 = session.getMap("Map2");
  Objectmap map3 = session.getMap("Map3");
  session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE);
  session.begin();
  map1.insert("randKey345", "HelloMap1");
  map2.insert("randKey58901", "HelloMap2");
  map3.insert("randKey58", "HelloMap3");
  session.commit();
  ```

Rubrique parent : 2.5+ Programmation de transactions dans des applications Java

Concepts associés:
Accès aux données et transactions

Java 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Java 2.5+

Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Si vous répartissez les données dans plusieurs partitions dans la grille de données, vous pouvez lire et mettre à jour plusieurs partitions dans une seule transaction. Ce type de transaction, appelée transaction multipartition, utilise le protocole de validation en deux phases pour coordonner et restaurer la transaction en cas d'échec.

2.5+ **Validation en deux phases et reprise sur incident**
Le protocole de validation en deux phases coordonne toutes les partitions qui participent à une transaction répartie, en déterminant si la transaction doit être validée ou annulée.

2.5+ **Développement d'applications pour écrire dans des transactions multipartitions pour WebSphere eXtreme Scale dans un environnement autonome**
Vous pouvez écrire une application pour une grille de données répartie avec plusieurs partitions dans l'environnement WebSphere eXtreme Scale autonome.

2.5+ **Développement de composants client eXtreme Scale en vue d'utiliser des transactions**
L'adaptateur de ressources WebSphere eXtreme Scale fournit une gestion des connexions client et une prise en charge des transactions locales. Ces prises en charge permettent aux applications Java Platform, Enterprise Edition (Java EE) de rechercher les connexions client eXtreme Scale et de démarquer les transactions locales à l'aide des transactions locales Java EE ou des API eXtreme Scale.

Rubrique parent : 2.5+ Programmation de transactions dans des applications Java

Tâches associées:
Java Interaction avec les données dans une transaction pour les applications Java

Java

Développement d'applications pour écrire dans des transactions multipartitions pour WebSphere eXtreme Scale dans un environnement autonome

2.5+ Vous pouvez écrire une application pour une grille de données répartie avec plusieurs partitions dans l'environnement WebSphere eXtreme Scale autonome.

Avant de commencer

Activez le protocole eXtremeIO. Pour plus d'informations, voir Configuration d'IBM eXtremeIO (XIO).

Restriction : Tenez compte des restrictions ci-dessous avant de développer des applications destinées à écrire dans des transactions multipartitions.

- Vous ne pouvez pas utiliser la réplication multimaître avec des transactions qui écrivent dans plusieurs partitions.
- Vous ne pouvez pas utiliser plusieurs partitions dans un client WebSphere eXtreme Scale dans une environnement .NET.
- Les mappes de sauvegarde configurées avec un plug-in de chargeur peuvent lire, mais pas écrire dans la mappe d'une transaction multipartition.
- Les mappes de sauvegarde qui utilisent une stratégie de verrouillage NONE ne peuvent pas participer aux transactions multipartitions.

Pourquoi et quand exécuter cette tâche

Utilisez l'API de session TxCommitProtocol définie pour activer le support de transaction multipartition pour WebSphere eXtreme Scale dans un environnement autonome. La nouvelle API fournit les deux options suivantes :

- TxCommitProtocol.ONEPHASE : constante du protocole de validation de transaction qui indique que la transaction doit être validée avec la validation en une étape par défaut. Avec cette option, une transaction peut lire plusieurs partitions, mais elle ne peut écrire que dans une seule partition. Une exception TransactionException se produit si la transaction écrit dans plusieurs partitions.
- TxCommitProtocol.TWOPHASE : constante du protocole de validation de transaction qui indique que la transaction doit être validée avec la validation en une ou deux phases. Si la transaction écrit dans une seule partition, le protocole de validation en une phase est utilisé. Dans le cas contraire, le protocole en deux phases est utilisé pour valider la transaction, impliquant des opérations d'écriture sur plusieurs partitions.

Vous pouvez configurer le support multitransaction pour WebSphere eXtreme Scale dans WebSphere Application Server. Pour plus d'informations, voir Développement de composants client eXtreme Scale en vue d'utiliser des transactions.

Procédure

1. Connectez-vous à la grille de données. Pour plus d'informations, voir Développement d'applications de grilles de données avec des API Java.
2. Obtenez une instance de session de la grille de données avec la méthode ObjectGrid.getSession. Pour plus d'informations, voir Développement d'applications de grilles de données avec des API Java.
3. Activez un protocole de validation en deux phases en définissant le fragment de code suivant : ²`session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE); session.begin();` Le fragment de code suivant montre comment créer, extraire, mettre à jour et supprimer des opérations dans une grille avec un protocole de validation en deux phases :

```
Session session = og.getSession();
Objectmap map1 = session.getMap("Map1");
Objectmap map2 = session.getMap("Map2");
Objectmap map3 = session.getMap("Map3");
session.setTxCommitProtocol(Session.TxCommitProtocol.TWOPHASE);
session.begin();
map1.insert("randKey345", "HelloMap1");
map2.insert("randKey58901", "HelloMap2");
map3.insert("randKey58", "HelloMap3");
session.commit();
```

Que faire ensuite

Vous pouvez activer la fonction de trace sur les transactions multipartitions. Pour plus d'informations, voir Analyse des données de journal et de trace.

Rubrique parent : 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Concepts associés:
Java Stratégies de verrouillage
Java Accès aux données et transactions

Java

Développement de composants client eXtreme Scale en vue d'utiliser des transactions

L'adaptateur de ressources WebSphere eXtreme Scale fournit une gestion des connexions client et une prise en charge des transactions locales. Ces prises en charge permettent aux applications Java™ Platform, Enterprise Edition (Java EE) de rechercher les connexions client eXtreme Scale et de démarquer les transactions locales à l'aide des transactions locales Java EE ou des API eXtreme Scale.

Avant de commencer

Créez une référence de ressource de fabrique de connexions eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser les API d'accès aux données eXtreme Scale de plusieurs façons. Dans tous les cas, la fabrique de connexions eXtreme Scale doit être injectée dans le composant d'application ou recherchée dans l'interface JNDI (Java Naming Directory Interface). Une fois la fabrique de connexions recherchée, vous pouvez démarquer les transactions et créer des connexions permettant d'accéder aux API eXtreme Scale.

Vous pouvez aussi éventuellement transtyper l'instance javax.resource.cci.ConnectionFactory en une fabrique com.ibm.websphere.xs.ra.XSConnectionFactory qui fournit des options supplémentaires pour l'extraction des descripteurs de connexion. Les descripteurs de connexions générés doivent être transtypés vers l'interface com.ibm.websphere.xs.ra.XSConnection, qui fournit la méthode getSession. La méthode getSession renvoie un descripteur d'objet com.ibm.websphere.objectgrid.Session qui permet aux applications d'utiliser n'importe laquelle des API d'accès aux données eXtreme Scale, telles que les API ObjectMap et EntityManager.

Le descripteur de session et les objets dérivés sont valides pendant toute la durée de vie du descripteur XSConnection.

Les procédures suivantes peuvent être utilisées pour démarquer les transactions eXtreme Scale. Vous ne pouvez pas mélanger ces procédures. Par exemple, vous ne pouvez pas mélanger une démarcation de transaction globale et une démarcation de transaction locale dans le même contexte de composant d'application.

Procédure

- Utilisez des transactions locales à validation automatique. Suivez les étapes ci-dessous pour utiliser automatiquement les opérations d'accès aux données à validation ou les opérations qui ne prennent pas en charge une transaction active :
    1. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection en dehors du contexte d'une transaction globale.
    2. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session pour interagir avec la grille de données.
    3. Appelez toute opération d'accès aux données prenant en charge les transactions à validation automatique.
    4. Fermez la connexion.
- Utilisez une session ObjectGrid pour démarquer une transaction locale. Suivez les étapes ci-dessous pour démarquer une transaction ObjectGrid à l'aide de l'objet de session :
    1. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection.
    2. Extrayez la session com.ibm.websphere.objectgrid.Session.
    3. Utilisez la méthode Session.begin() pour démarrer la transaction.
    4. Utilisez la session pour interagir avec la grille de données.
    5. Utilisez les méthodes Session.commit() ou rollback() pour mettre fin à la transaction.
    6. Fermez la connexion.
- Utilisez une transaction javax.resource.cci.LocalTransction pour démarquer une transaction locale. Suivez les étapes ci-dessous pour démarquer une transaction ObjectGrid à l'aide de l'interface javax.resource.cci.LocalTransaction :
    1. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection.
    2. Extrayez la transaction javax.resource.cci.LocalTransaction à l'aide de la méthode XSConnection.getLocalTransaction().
    3. Utilisez la méthode LocalTransaction.begin() pour démarrer la transaction.

4. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session pour interagir avec la grille de données.
5. Utilisez les méthodes LocalTransaction.commit() ou rollback() pour mettre fin à la transaction.
6. Fermez la connexion.

- Inscrivez la connexion dans une transaction globale. Cette procédure s'applique également aux transactions gérées par conteneur :
  1. Commencez la transaction globale à l'aide de l'interface javax.transaction.UserTransaction ou d'une transaction gérée par conteneur.
  2. Extrayez une connexion com.ibm.websphere.xs.ra.XSConnection.
  3. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session.
  4. Fermez la connexion.
  5. Validez ou annulez la transaction globale.
- 2.5+ Configurez une connexion pour écrire plusieurs partitions dans une transaction. Suivez les étapes ci-dessous pour démarquer une transaction ObjectGrid à l'aide de l'objet de session :
  1. Créez un objet com.ibm.websphere.xs.ra.XSConnectionSpec.
  2. Appelez la méthode XSConnectionSpec et la méthode setMultiPartitionSupportEnabled avec l'argument `true`.
  3. Extrayez la connexioncom.ibm.websphere.xs.ra.XSConnection pour envoyer XSConnectionSpec à la méthode ConnectionFactory.getConnection.
  4. Extrayez et utilisez la session com.ibm.websphere.objectgrid.Session.

Exemple

Reportez-vous à l'exemple de code suivant, qui illustre les étapes précédentes de démarcation des transactions eXtreme Scale.

```
    // (C) Copyright IBM Corp. 2001, 2012.
    // All Rights Reserved. Eléments sous licence - Propriété d'IBM.
    package com.ibm.ws.xs.ra.test.ee;

    import javax.naming.InitialContext;
    import javax.resource.cci.Connection;
    import javax.resource.cci.ConnectionFactory;
    import javax.resource.cci.LocalTransaction;
    import javax.transaction.Status;
    import javax.transaction.UserTransaction;

    import junit.framework.TestCase;

    import com.ibm.websphere.objectgrid.ObjectMap;
    import com.ibm.websphere.objectgrid.Session;
    import com.ibm.websphere.xs.ra.XSConnection;

    /**
    * Cet exemple doit être exécuté dans un contexte J2EE sur votre serveur
    * d'applications. Par exemple, en utilisant le servlet d'infrastructure préfabriquée
JUnitEE.
    *
    * Le code présent dans ces méthodes de texte réside généralement sur votre propre
servlet,
    * sur votre EJB ou sur un autre composant Web.
    *
    * L'exemple dépend d'une fabrique de connexions WebSphere eXtreme Scale configurée
    * enregistrée sous le nom JNDI de "eis/embedded/wxscf" qui définit une
    * connexion à une grille contenant une mappe portant le nom "Map1".
    *
    * Cet exemple effectue une recherche directe du nom JNDI et ne nécessite pas une
    * injection de ressource.
    */
    public class DocSampleTests extends TestCase {
        public final static String CF_JNDI_NAME = "eis/embedded/wxscf";
        public final static String MAP_NAME = "Map1";

        Long                key = null;
```

```
        Long                value = null;
        InitialContext      ctx = null;
        ConnectionFactory   cf = null;

        public DocSampleTests() {
        }
        public DocSampleTests(String name) {
            super(name);
        }
        protected void setUp() throws Exception {
            ctx = new InitialContext();
            cf = (ConnectionFactory)ctx.lookup(CF_JNDI_NAME);
            key = System.nanoTime();
            value = System.nanoTime();
        }
        /**
         * Cette exemple s'exécute lorsqu'il n'est pas dans le contexte d'une transaction
globale
         * et utilise la validation automatique (autocommit).
         */
        public void testLocalAutocommit() throws Exception {
            Connection conn = cf.getConnection();
            try {
                Session    session = ((XSConnection)conn).getSession();
                ObjectMap  map = session.getMap(MAP_NAME);
                map.insert(key, value); // Or various data access operations
            }
            finally {
                conn.close();
            }
        }

        /**
         * Cette exemple s'exécute lorsqu'il n'est pas dans le contexte d'une transaction
globale
         * et démarque la transaction à l'aide de session.begin()/session.commit()
         */
        public void testLocalSessionTransaction() throws Exception {
            Session    session = null;
            Connection conn = cf.getConnection();
            try {
                session = ((XSConnection)conn).getSession();
                session.begin();
                ObjectMap  map = session.getMap(MAP_NAME);
                map.insert(key, value); // Or various data access operations
                session.commit();
            }
            finally {
                if (session != null && session.isTransactionActive()) {
                    try { session.rollback(); }
                    catch (Exception e) { e.printStackTrace(); }
                }
                conn.close();
            }
        }

        /**
         * Cet exemple utilise l'interface LocalTransaction pour démarquer les
         * transactions.
         */
        public void testLocalTranTransaction() throws Exception {
            LocalTransaction tx = null;
            Connection conn = cf.getConnection();
            try {
                tx = conn.getLocalTransaction();
                tx.begin();
                Session    session = ((XSConnection)conn).getSession();
                ObjectMap  map = session.getMap(MAP_NAME);
```

```
            map.insert(key, value); // Or various data access operations
            tx.commit(); tx = null;
        }
        finally {
            if (tx != null) {
                try { tx.rollback(); }
                catch (Exception e) { e.printStackTrace(); }
            }
            conn.close();
        }
    }

    /**
     * Cet exemple dépend d'une transaction à gestion externe.
     * Cette transaction à gestion externe peut généralement être présente dans
     * un EJB avec des attributs de transaction définis sur REQUIRED ou REQUIRES_NEW.
     * REMARQUE : S'il n'y a AUCUNE transaction globale active, cet exemple s'exécute
en
     *       mode de validation automatique car il ne vérifie pas si une transaction
existe.
     */
    public void testGlobalTransactionContainerManaged() throws Exception {
        Connection      conn = cf.getConnection();
        try {
            Session    session = ((XSConnection)conn).getSession();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
        }
        catch (Throwable t) {
            t.printStackTrace();
            UserTransaction tx =
(UserTransaction)ctx.lookup("java:comp/UserTransaction");
            if (tx.getStatus() != Status.STATUS_NO_TRANSACTION) {
                tx.setRollbackOnly();
            }
        }
        finally {
            conn.close();
        }
    }

    /**
     * Cet exemple montre comment démarrer une nouvelle transaction globale à l'aide
de
     * l'interface UserTransaction. Généralement, le conteneur démarre la transaction
     * globale (par exemple, dans un EJB avec un attribut de transaction défini sur
     * REQUIRES_NEW), mais cet exemple démarre aussi la transaction globale
     * à l'aide de l'API UserTransaction si elle n'est pas actuellement active.
     */
    public void testGlobalTransactionTestManaged() throws Exception {
        boolean         started = false;
        UserTransaction tx = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
        if (tx.getStatus() == Status.STATUS_NO_TRANSACTION) {
            tx.begin();
            started = true;
        }
        // else { called with an externally/container managed transaction }
        Connection conn = null;
        try {
            conn = cf.getConnection(); // Get connection after the global tran starts
            Session    session = ((XSConnection)conn).getSession();
            ObjectMap  map = session.getMap(MAP_NAME);
            map.insert(key, value); // Or various data access operations
            if (started) {
                tx.commit(); started = false; tx = null;
            }
        }
        finally {
```

```
                if (started) {
                    try { tx.rollback(); }
                    catch (Exception e) { e.printStackTrace(); }
                }
                if (conn != null) { conn.close(); }
            }
        }
 /**
         /**
         * Cet exemple montre un transaction multipartition.
         */

                                public void
testGlobalTransactionTestManagedMultiPartition() throws Exception {
                                                boolean         started = false;
                                                XSConnectionSpec connSpec = new
XSConnectionSpec();

connSpec.setWriteToMultiplePartitions(true);
                                                UserTransaction tx =
(UserTransaction)ctx.lookup("java:comp/UserTransaction");
                                                if (tx.getStatus() ==
Status.STATUS_NO_TRANSACTION) {
                                                                tx.begin();
                                                started = true;
                                        }
                                // else { called with an externally/container managed
transaction }
                                                Connection conn = null;
                                                                        try {
                                                conn = cf.getConnection(connSpec); // Get
connection after the global tran starts
                                                Session    session =
((XSConnection)conn).getSession();
                                                ObjectMap  map = session.getMap(MAP_NAME);
                                                map.insert(key, value); // Or various data
access operations
                                                if (started) {
                                                                tx.commit(); started =
false; tx = null;
                                                                }
                                                }
                                                finally {
                                                        if (started) {
                                                                        try {
tx.rollback(); }
                                                                        catch (Exception
e) { e.printStackTrace(); }
                                                        }
                                                        if (conn != null) { conn.close();
}
                                                }
                                        }
```

Rubrique parent : 2.5+ Développement d'applications qui mettent à jour plusieurs partitions dans une seule transaction

Utilisation du verrouillage

Les verrous comportent des cycles de vie et leurs différents types sont compatibles entre eux selon plusieurs critères. Les verrous doivent être traités dans un ordre approprié pour éviter les situations d'interblocage.

2.5+ **Configuration et mise en oeuvre du verrouillage dans les applications Java**
Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale.

2.5+ **Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java**
Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Java 2.5+ **Exemple : ordonnancement des verrous avec la méthode flush**
L'appel de la méthode flush sur l'interface ObjectMap avant une validation peut introduire des contraintes supplémentaires concernant l'ordre des verrous. La méthode flush est généralement utilisée pour forcer la transmission des modifications apportées à la mappe au programme d'arrière-plan par l'intermédiaire du plug-in du chargeur.

Java 2.5+ **Exemples Java pour l'isolement de transaction**
L'isolement de transaction définit comment les modifications apportées par une opération deviennent visibles aux autres opérations simultanées. Vous pouvez utiliser les exemples suivants pour définir le niveau d'isolement de transaction dans votre application Java.

Rubrique parent : 2.5+ Programmation de transactions dans des applications Java

Java

Configuration et mise en oeuvre du verrouillage dans les applications Java

Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque mappe de sauvegarde (BackingMap) de la configuration de WebSphere eXtreme Scale.

Avant de commencer

- Déterminez la stratégie de verrouillage à utiliser. Pour plus d'informations, voir Stratégies de verrouillage.
- Vous pouvez également configurer une stratégie de verrouillage en configurant une mappe dynamique. Pour plus d'informations, voir Configuration d'une stratégie de verrouillage.

Pourquoi et quand exécuter cette tâche

Pour éviter une exception java.lang.IllegalStateException, vous devez appeler la méthode setLockStrategy avant d'appeler les méthodes initialize ou getSession sur l'instance ObjectGrid.

Procédure

1. Configurez une stratégie de verrouillage dans votre application Java™.
   - Configurez une stratégie de verrouillage optimiste. Utilisez pour cela la méthode setLockStrategy :

     ```
     import com.ibm.websphere.objectgrid.BackingMap;
     import com.ibm.websphere.objectgrid.LockStrategy;
     import com.ibm.websphere.objectgrid.ObjectGrid;
     import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
     ...
     ObjectGrid og =
             ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test"
     );
     BackingMap bm = og.defineMap("optimisticMap");
     bm.setLockStrategy( LockStrategy.OPTIMISTIC );
     ```

   - Configurez une stratégie de verrouillage pessimiste. Utilisez pour cela la méthode setLockStrategy :

     ```
     import com.ibm.websphere.objectgrid.BackingMap;
     import com.ibm.websphere.objectgrid.LockStrategy;
     import com.ibm.websphere.objectgrid.ObjectGrid;
     import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
     ...
     ObjectGrid og =
             ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test"
     );
     BackingMap bm = og.defineMap("pessimisticMap");
     bm.setLockStrategy( LockStrategy.PESSIMISTIC);
     ```

   - Configurez une stratégie sans verrouillage. Utilisez pour cela la méthode setLockStrategy :

     Remarque : Les mappes de sauvegarde configurées pour utiliser une stratégie sans verrouillage ne peuvent pas participer à une transaction multipartition.

     ```
     import com.ibm.websphere.objectgrid.BackingMap;
     import com.ibm.websphere.objectgrid.LockStrategy;
     import com.ibm.websphere.objectgrid.ObjectGrid;
     import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
     ...
     ObjectGrid og =
             ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test"
     );
     BackingMap bm = og.defineMap("noLockingMap");
     bm.setLockStrategy( LockStrategy.NONE);
     ```

2. Configurez un délai de verrouillage. Utilisez la méthode setLockTimeout sur l'instance BackingMap :

```
bm.setLockTimeout( 60 );
```

Le paramètre de la méthode setLockTimeout est un entier primitif Java qui définit le délai en secondes pendant lequel eXtreme Scale attend l'octroi d'un mode de verrouillage. Si le temps d'attente d'une transaction excède le délai d'attente de verrouillage configurée pour la mappe de sauvegarde, une exception com.ibm.websphere.objectgrid.LockTimeoutException est émise.

3. Si vous utilisez une stratégie de verrouillage pessimiste, vous pouvez utiliser la méthode lock pour verrouiller la clé dans la grille de données, ou verrouiller la clé et déterminer si la valeur existe dans la grille de données. Dans les éditions précédentes, vous utilisiez les API get et getForUpdate pour verrouiller des clés dans la grille de données. Toutefois, si vous n'aviez pas besoin des données du client, les performances étaient dégradées lors de l'extraction des objets de valeur potentiellement volumineux vers le client. De plus, la méthode containsKey ne maintenant aucun verrou, vous étiez obligé d'utiliser les méthodes get et getForUpdate pour obtenir les verrous appropriés en cas d'utilisation du verrouillage pessimiste. L'API lock fournit désormais une méthode containsKey lors du maintien du verrou. Etudiez les exemples suivants :
   - Les méthodes suivantes verrouillent la clé dans la mappe et renvoient la valeur true si la clé existe ou la valeur false si la clé n'existe pas :

     ```
     boolean ObjectMap.lock(Object key, LockMode lockMode);
     ```

   - La méthode suivante verrouille une liste de clés dans la mappe et renvoie une liste de valeurs true ou false ; `true` si la clé existe, `false` si elle n'existe pas :

     ```
     List<Boolean> ObjectMap.lockAll(List keys, LockMode lockMode);
     ```

     LockMode est une énumération qui vous permet d'indiquer les clés que vous souhaitez verrouiller à l'aide des valeurs possibles suivantes :
     - Partagé (SHARED), pouvant être mis à niveau (UPGRADEABLE) et exclusif (EXCLUSIVE)

Voici un exemple de définition du paramètre LockMode :

```
session.begin();
map.lock(key, LockMode.UPGRADABLE);
map.upsert();
session.commit()
```

Rubrique parent : 2.5+ Utilisation du verrouillage

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
Configuration d'une stratégie de verrouillage

Référence associée:
Exemple : ordonnancement des verrous avec la méthode flush

Java

Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java™

Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Procédure

1. Intercepter l'exception et afficher le message résultant.

```
try {
...
} catch (ObjectGridException oe) {
System.out.println(oe);
}
```

L'exception suivante s'affiche :

```
com.ibm.websphere.objectgrid.plugins.LockDeadlockException: Message
```

Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

2. Annuler la transaction après une exception :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
boolean activeTran = false;
try
{
    sess.begin();
    activeTran = true;
    Person p = (IPerson)person.get("Lynn");
    // Lynn a fêté son anniversaire, donc nous l'avons vieilli d'1 an.
    p.setAge( p.getAge() + 1 );
    person.put( "Lynn", p );
    sess.commit();
    activeTran = false;
}
finally
{
    if ( activeTran ) sess.rollback();
}
```

Le bloc `finally` dans le fragment de code garantit qu'une transaction est annulée lorsqu'une exception inattendue se produit. Il ne gère pas seulement une exception de type LockDeadlockException mais également les autres exceptions imprévues éventuelles. Il traite le cas où une exception est émise lors d'un appel de la méthode commit. Cet exemple ne constitue pas le seul moyen pour traiter les exceptions inattendues ; il existe des cas où une application souhaite intercepter certaines des exceptions inattendues susceptibles de se produire afin de pouvoir afficher l'une de ses exceptions d'application. Vous pouvez ajouter des blocs catch comme il convient mais l'application doit faire en sorte que le fragment de code ne se ferme pas sans terminer la transaction.

Rubrique parent : 2.5+ Utilisation du verrouillage

Java

Exemple : ordonnancement des verrous avec la méthode flush

L'appel de la méthode flush sur l'interface ObjectMap avant une validation peut introduire des contraintes supplémentaires concernant l'ordre des verrous. La méthode flush est généralement utilisée pour forcer la transmission des modifications apportées à la mappe au programme d'arrière-plan par l'intermédiaire du plug-in du chargeur.

Dans ce cas, le programme d'arrière-plan utilise son propre gestionnaire de verrou pour contrôler les accès simultanés, ce qui fait que l'état d'attente de verrou et l'interblocage peuvent se produire dans le programme d'arrière-plan et non dans le gestionnaire de verrou de WebSphere eXtreme Scale Client. Examinons la transaction suivante :

```
Session sess = ...;
ObjectMap person = sess.getMap("PERSON");
boolean activeTran = false;
try
{
    sess.begin();
    activeTran = true;
    Person p = (IPerson)person.get("Lynn");
    p.setAge( p.getAge() + 1 );
    person.put( "Lynn", p );
    person.flush();
    ...
    p = (IPerson)person.get("Tom");
    p.setAge( p.getAge() + 1 );
    sess.commit();
    activeTran = false;
}
finally
{
    if ( activeTran ) sess.rollback();
}
```

Supposons qu'une autre transaction a également mis à jour la personne Tom, a appelé la méthode flush, puis a mis à jour la personne Lynn. Si cette situation se présente, l'entrelacement ci-dessous des deux transactions entraîne une condition d'interblocage de la base de données :

```
Verrou X octroyé à la transaction 1 pour "Lynn" lorsque la méthode flush est exécutée.
Verrou X octroyé à la transaction 2 pour "Tom" lorsque la méthode flush est exécutée.
Verrou X demandé par la transaction 1 pour "Tom" pendant le traitement de la validation.
(La transaction 1 se bloque en attente du verrou acquis par la transaction 2.)
Verrou X demandé par la transaction 2 pour "Lynn" pendant le traitement de la validation.
(La transaction 2 se bloque en attente du verrou acquis par la transaction 1.)
```

Cet exemple montre que l'utilisation de la méthode flush peut provoquer un interblocage dans la base de données plutôt que dans WebSphere eXtreme Scale Client. Cet exemple d'interblocage peut se produire indépendamment de la stratégie de verrouillage utilisée. L'application doit veiller à empêcher ce type d'interblocage lors de l'utilisation de la méthode flush et lorsqu'un chargeur est connecté à la mappe de sauvegarde. L'exemple précédent illustre également pourquoiWebSphere eXtreme Scale Client comporte un mécanisme de délai d'attente de verrou. Une transaction qui attend un verrou de base de données peut patienter alors qu'elle possède un verrou d'entrée de mappe WebSphere eXtreme Scale Client. Les problèmes rencontrés au niveau de la base de données peuvent causer des temps d'attente excessifs pour un mode de verrouillage WebSphere eXtreme Scale Client et entraîner l'émission d'une exception LockTimeoutException.

Rubrique parent : 2.5+ Utilisation du verrouillage

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
Configuration d'une stratégie de verrouillage
Java Configuration et mise en oeuvre du verrouillage dans les applications Java

Java

Exemples Java pour l'isolement de transaction

L'isolement de transaction définit comment les modifications apportées par une opération deviennent visibles aux autres opérations simultanées. Vous pouvez utiliser les exemples suivants pour définir le niveau d'isolement de transaction dans votre application Java™.

Lecture reproductible avec verrouillage pessimiste

```
map = session.getMap("Order");
session.setTransactionIsolation(Session.TRANSACTION_REPEATABLE_READ);
session.begin();

// Un verrou S est demandé et maintenu et la valeur est copiée dans
// le cache transactionnel.
Order order = (Order) map.get("100");
// L'entrée est expulsée du cache transactionnel.
map.invalidate("100", false);

// La même valeur est demandée de nouveau. Elle contient déjà le
// verrou, donc la même valeur est extraite et copiée dans le
// cache transactionnel.
Order order2 = (Order) map.get("100");

// Tous les verrous sont annulés après la synchronisation de la transaction
// avec la mappe de cache.
session.commit();
```

Lecture validée avec verrouillage pessimiste

```
map1 = session1.getMap("Order");
session1.setTransactionIsolation(Session.TRANSACTION_READ_COMMITTED);
session1.begin();

// Un verrou S est demandé mais immédiatement annulé et
//la valeur est copiée dans le cache transactionnel.

Order order = (Order) map1.get("100");

// L'entrée est expulsée du cache transactionnel.
map1.invalidate("100", false);

// Une deuxième transaction met à jour la même commande.
// Elle obtient un verrou U, met à jour la valeur et valide.
// L'ObjectGrid obtient correctement le verrou X lors de la
// validation car la première transaction utilise l'isolement lecture
// validée.

Map orderMap2 = session2.getMap("Order");
session2.begin();
order2 = (Order) orderMap2.getForUpdate("100");
order2.quantity=2;
orderMap2.update("100", order2);
session2.commit();

// La même valeur est demandée de nouveau. Cette fois, la valeur doit être
// mise à jour mais elle reflète maintenant la
// nouvelle valeur
Order order1Copy = (Order) map1.getForUpdate("100");
```

Rubrique parent : 2.5+ Utilisation du verrouillage

Concepts associés:
Isolement des transactions

Java

Plug-in d'indexation des données

Selon le type d'index que vous voulez générer, WebSphere eXtreme Scale Client fournit des plug-in intégrés que vous pouvez ajouter à la mappe de sauvegarde pour générer un index.

HashIndex

Le plug-in HashIndex intégré, la classe com.ibm.websphere.objectgrid.plugins.index.HashIndex, est un plug-in MapIndexPlugin que vous pouvez ajouter à la mappe de sauvegarde pour générer des index dynamiques. Cette classe prend en charge à la fois les interfaces MapIndex et MapRangeIndex. La définition et l'implémentation d'index peuvent considérablement améliorer les performances des requêtes.

Accès aux données avec des index (API Index)
Utilisez l'indexation pour améliorer l'accès aux données.

Utilisation des sessions pour accéder aux données de la grille
Les applications peuvent commencer et terminer les transactions par le biais de l'interface Session. L'interface Session permet également d'accéder aux interfaces ObjectMap et JavaMap d'application.

Configuration du plug-in HashIndex
Vous pouvez configurer le plug-in HashIndex intégré (classe com.ibm.websphere.objectgrid.plugins.index.HashIndex) à l'aide d'un programme, ou à l'aide d'un index dynamique.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Java

Accès aux données avec des index (API Index)

Utilisez l'indexation pour améliorer l'accès aux données.

Pourquoi et quand exécuter cette tâche

La classe HashIndex est l'implémentation de plug-in d'indexation intégré qui peut prendre en charge les deux interfaces d'indexation d'application intégrée MapIndex et MapRangeIndex. Vous pouvez également créer vos propres index. Vous pouvez ajouter HashIndex sous la forme d'un index dynamique dans la mappe de sauvegarde, obtenir soit un objet proxy d'index MapIndex ou MapRangeIndex et utiliser cet objet proxy d'index pour rechercher des objets mis en cache.

Si vous souhaitez effectuer une itération dans les clés d'une mappe locale, vous pouvez utiliser l'index par défaut. Cet index ne requiert pas de configuration, mais il doit être utilisé sur le fragment en utilisant un agent ou une instance ObjectGrid extraits de la méthode ShardEvents.shardActivated(ObjectGrid shard).

Remarque : Dans un environnement réparti, si l'objet d'index est obtenu à partir d'un ObjectGrid client, l'index possède un objet de type client et toutes les opérations d'index sont exécutées dans un ObjectGrid de serveur. Si la mappe est partitionnée, les opérations d'indexation sont exécutées dans chaque partition à distance. Les résultats de chaque partition sont fusionnés avant de renvoyer les résultats à l'application. Les performances sont déterminées par le nombre de partitions et la taille des résultats renvoyés par chaque partition. Les performances peuvent être faibles si les deux facteurs sont élevés.

Procédure

Accédez aux clés de mappe et aux valeurs avec des index.

- **Index local :**

  Pour effectuer une itération dans les clés d'une mappe locale, vous pouvez utiliser l'index par défaut. Cet index fonctionne uniquement avec le fragment, en utilisant un agent ou l'instance ObjectGrid extraits de la méthode ShardEvents.shardActivated(ObjectGrid shard). Reportez-vous à l'exemple suivant :

  ```
  MapIndex keyIndex = (MapIndex)
  objMap.getIndex(MapIndexPlugin.SYSTEM_KEY_INDEX_NAME);
  Iterator keyIterator = keyIndex.findAll();
  ```

- **Index dynamiques :**

  Vous pouvez créer et supprimer à tout moment des index dynamiques d'une instance BackingMap. Un index dynamique diffère d'un index statique en ce qu'il peut être créé même après l'initialisation de l'instance ObjectGrid contenante. Contrairement à l'indexation statique, l'indexation dynamique est un processus asynchrone et doit être à l'état prêt (ready) avant d'être utilisée. Cette méthode utilise la même approche pour extraire et utiliser les index dynamiques que pour les index statiques. Vous pouvez supprimer un index dynamique s'il n'est plus utile. L'interface BackingMap comprend deux méthodes pour créer et supprimer des index dynamiques.

  Pour plus d'informations concernant les méthodes createDynamicIndex et removeDynamicIndex, voir API BackingMap.

  ```
  import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
  import com.ibm.websphere.objectgrid.ObjectGridManager;
  import com.ibm.websphere.objectgrid.ObjectGrid;
  import com.ibm.websphere.objectgrid.BackingMap;

         ObjectGridManager ogManager =
  ObjectGridManagerFactory.getObjectGridManager();
                BackingMap bm = og.getMap("person");

         // créez l'index après l'initialisation de l'ObjectGrid sans
  DynamicIndexCallback.
         bm.createDynamicIndex("CODE", true, "employeeCode", null);

         try {
             // Si DynamicIndexCallback n'est pas utilisé, attendre que l'index soit
  prêt.
             // Le délai d'attente dépend de la taille actuelle de la mappe
             Thread.sleep(3000);
         } catch (Throwable t) {
  ```

```
            // ...
        }

        // Lorsque l'index est prêt, les applications peuvent tenter d'obtenir
l'instance d'interface
        // de l'index d'application.
        // Les applications doivent trouver un moyen de vérifier que l'index est
prêt à être utilisé,
        // faute de quoi elles utilisent l'interface DynamicIndexCallback.
        // L'exemple ci-dessous illustre comment attendre que l'index soit prêt
        // Prenez en compte la taille de la mappe dans le délai d'attente total.

        Session session = og.getSession();
        ObjectMap m = session.getMap("person");
        MapRangeIndex codeIndex = null;

        int counter = 0;
        int maxCounter = 10;
        boolean ready = false;
        while (!ready && counter < maxCounter) {
            try {
                counter++;
                codeIndex = (MapRangeIndex) m.getIndex("CODE");
                ready = true;
            } catch (IndexNotReadyException e) {
                // implique que l'index n'est pas prêt, ...
                System.out.println("Index non prêt. Veuillez patienter.");
                try {
                    Thread.sleep(3000);
                } catch (Throwable tt) {
                    // ...
                }
            } catch (Throwable t) {
                // exception inattendue
                t.printStackTrace();
            }
        }

        if (!ready) {
            System.out.println("Index non prêt. Remédier à cette situation.");
        }

        // Utilisez l'index pour exécuter des requêtes
        // Reportez-vous à l'interface MapIndex ou MapRangeIndex pour les opérations
prises en charge.
        // L'attribut d'objet sur lequel repose l'index est EmployeeCode.
        // Supposons que l'attribut EmployeeCode est de type entier : le
        // paramètre transmis aux opérations d'index présente ce type de données.

        Iterator iter = codeIndex.findLessEqual(new Integer(15));

        // supprimez l'index dynamique lorsqu'il n'est plus nécessaire

        bm.removeDynamicIndex("CODE");
        // Fermer la session (facultatif dans les versions 7.1.1 et ultérieures)
pour améliorer les performances
        session.close();
```

Que faire ensuite

Vous pouvez utiliser l'interface DynamicIndexCallback pour obtenir des notifications des événements d'indexation. Pour plus d'informations, voir Interface DynamicIndexCallback.

Interface DynamicIndexCallback
L'interface DynamicIndexCallback est destinée aux applications qui veulent recevoir des notifications pour les événements d'indexation ready, error ou destroy. DynamicIndexCallback est un paramètre facultatif pour la méthode createDynamicIndex de la mappe de sauvegarde. Avec une instance enregistrée DynamicIndexCallback, les applications peuvent exécuter la logique applicative à la

réception de la notification d'un événement d'indexation.

Rubrique parent : Plug-in d'indexation des données

Java

Interface DynamicIndexCallback

L'interface DynamicIndexCallback est destinée aux applications qui veulent recevoir des notifications pour les événements d'indexation ready, error ou destroy. DynamicIndexCallback est un paramètre facultatif pour la méthode createDynamicIndex de la mappe de sauvegarde. Avec une instance enregistrée DynamicIndexCallback, les applications peuvent exécuter la logique applicative à la réception de la notification d'un événement d'indexation.

Evénements d'indexation

Par exemple, l'événement ready signifie que l'index est prêt à être utilisé. Lorsqu'une notification est reçue pour cet événement, une application peut tenter d'extraire et d'utiliser l'instance d'interface de l'index d'application.

Exemple : utilisation de l'interface DynamicIndexCallback

```
BackingMap personBackingMap = ivObjectGrid.getMap("person");
    DynamicIndexCallback callback = new DynamicIndexCallbackImpl();
    personBackingMap.createDynamicIndex("CODE", true, "employeeCode", callback);


    class DynamicIndexCallbackImpl implements DynamicIndexCallback {
        public DynamicIndexCallbackImpl() {
        }

        public void ready(String indexName) {
            System.out.println("DynamicIndexCallbackImpl.ready() -> indexName = " +
indexName);

            // Simulez le comportement d'une application lors de la notification ready.
            // Normalement, l'application doit patienter jusqu'à l'obtention de l'état
ready, puis doit mettre en oeuvre
            // la logique d'utilisation de l'index.
            if("CODE".equals(indexName)) {
                ObjectGridManager ogManager =
ObjectGridManagerFactory.getObjectGridManager();
                ObjectGrid og = ogManager.createObjectGrid( "grid" );
                Session session = og.getSession();
                ObjectMap map = session.getMap("person");
                MapIndex codeIndex = (MapIndex) map.getIndex("CODE");
                Iterator iter = codeIndex.findAll(codeValue);
                // Fermer la session (facultatif dans les versions 7.1.1 et ultérieures)
pour améliorer les performances
                session.close();
                                    }
        }

        public void error(String indexName, Throwable t) {
            System.out.println("DynamicIndexCallbackImpl.error() -> indexName = " +
indexName);
            t.printStackTrace();
        }

        public void destroy(String indexName) {
            System.out.println("DynamicIndexCallbackImpl.destroy() -> indexName = " +
indexName);
        }
    }
```

Rubrique parent : Accès aux données avec des index (API Index)

Java

Utilisation des sessions pour accéder aux données de la grille

Les applications peuvent commencer et terminer les transactions par le biais de l'interface Session. L'interface Session permet également d'accéder aux interfaces ObjectMap et JavaMap d'application.

Chaque instance ObjectMap ou JavaMap est directement associée à un objet Session spécifique. Chaque unité d'exécution souhaitant accéder à eXtreme Scale doit préalablement obtenir une instance Session à partir de l'objet ObjectGrid. Une instance Session ne peut pas être partagée par plusieurs unités d'exécution. WebSphere eXtreme Scale n'utilise aucun stockage local d'unités d'exécution ; cependant, les restrictions de plate-forme risquent de limiter le passage d'une Session d'une unité d'exécution à une autre.

Méthodes

Méthode Get

Une application obtient une instance Session à partir d'un objet ObjectGrid à l'aide de la méthode ObjectGrid.getSession. L'exemple suivant illustre comment obtenir une instance Session :

```
ObjectGrid objectGrid = ...; Session sess = objectGrid.getSession();
```

Une fois l'instance Session obtenue, l'unité d'exécution garde une référence à la session pour son propre usage. L'appel de la méthode getSession plusieurs fois renvoie un nouvel objet Session chaque fois.

Transactions et méthodes de session

Une session peut être utilisée pour commencer, valider ou annuler une transaction. Les opérations sur BackingMaps avec ObjectMaps et JavaMaps sont exécutées plus efficacement dans une transaction Session. Une fois une transaction commencée, toute modification apportée à un ou plusieurs mappes de sauvegarde dans cette transaction est stockée dans un cache de transaction spécial jusqu'à ce que la transaction soit validée. Lorsqu'une transaction est validée, les modifications en attente sont appliquées aux BackingMaps et Loaders et deviennent visibles par tous les clients de cet ObjectGrid.

WebSphere eXtreme Scale permet également de valider automatiquement les transactions. Si une opération ObjectMap est effectuée en dehors du contexte d'une transaction active, une transaction implicite est lancée avant l'opération et la transaction est automatiquement validée avant de rendre le contrôle à l'application.

```
Session session = objectGrid.getSession();
ObjectMap objectMap = session.getMap("someMap");
session.begin();
objectMap.insert("key1", "value1");
objectMap.insert("key2", "value2");
session.commit();
cobjectMap.insert("key3", "value3"); // auto-validation
```

Méthode Session.flush

La méthode Session.flush est utile lorsqu'un chargeur est associé à une BackingMap. La méthode flush appelle le chargeur avec l'ensemble de modifications actuel dans le cache de transaction. Le chargeur applique les changements au dorsal. Les changements ne sont pas validés lorsque la méthode flush est appelée. Si une transaction de session est validée après l'appel de flush, seules les mises à jour postérieures à l'appel de flush sont appliquées au chargeur. Si une transaction de session est annulée après l'appel de la méthode flush, les modifications vidées sont annulées, de même que toutes les autres modifications en attente dans la transaction. Utilisez la méthode flush avec parcimonie car elle limite les opérations de traitement par lots pour un chargeur. L'exemple ci-dessous illustre l'utilisation de la méthode Session.flush :

```
Session session = objectGrid.getSession();
session.begin();
// faites des modifications supplémentaires
...
session.flush(); // envoyez ces modifications vers le programme de chargement, mais
// ne validez pas ces modifications supplémentaires
...
session.commit();
```

Méthode NoWriteThrough

Certaines cartes sont sauvegardées par un chargeur, qui fournit un stockage de persistance aux données dans la mappe. Il est parfois utile de valider les données uniquement dans la mappe eXtreme Scale et de ne pas envoyer les données au chargeur. L'interface de session fournit la méthode beginNoWriteThough dans ce

but. La méthode beginNoWriteThrough commence une transaction comme la méthode begin. Avec la méthode beginNoWriteThrough, lorsque la transaction est validée, les données sont uniquement validées dans la mappe mémoire et elles ne sont pas validées dans le stockage de persistance fourni par le chargeur. Cette méthode est particulièrement utile lors du préchargement des données sur la mappe.

Lors de l'utilisation d'une instance ObjectGrid répartie, la méthode beginNoWriteThrough est utile pour effectuer des modifications dans le cache proche uniquement, sans modifier le cache éloigné du serveur. Si les données sont périmées dans le cache proche, l'utilisation de la méthode beginNoWriteThrough peut permettre d'invalider les entrées sur le cache proche sans les invalider sur le serveur.

L'interface Session fournit également la méthode isWriteThroughEnabled pour déterminer quel type de transaction est actuellement actif.

```
Session session = objectGrid.getSession();
session.beginNoWriteThrough();
// faites des modifications supplémentaires...
session.commit(); // ces modifications ne seront pas envoyées au chargeur
```

Obtention de la méthode d'objet TxID

L'objet TxID est un objet opaque qui identifie la transaction active. Utilisez l'objet TxID pour les applications suivantes :

- Pour effectuer des comparaisons lorsque vous recherchez une transaction spécifique.
- Pour stocker des données partagées entre les objets TransactionCallback et Loader.
- Déterminez si la transaction a été lancée à partir d'une transaction de session qui utilisait un protocole de validation en une ou deux phases. En examinant la sortie TxID.toString(), vous pouvez déterminer si la transaction était destinée à une transaction monopartition ou multipartition. Si la chaîne commence par le mot clé "Local", cela indique qu'il s'agit d'une transaction à une seule partition. Par exemple : `Local-40000139-72B2-C037-E000-1C271366B073` Si la chaîne commence par le mot clé "WXS", cela indique qu'il s'agit d'une transaction multipartition. Par exemple : `WXS-40000139-72B2-BD3A-E000-1C271366B073`

Reportez-vous au plug-in TransactionCallback et aux Loaders pour des informations supplémentaires sur la fonction d'attribut Object.

Méthode de suivi des performances

Si vous utilisez eXtreme Scale dans WebSphere Application Server, il peut être nécessaire de réinitialiser le type de transaction pour le suivi des performances. Vous pouvez définir le type de transaction avec la méthode setTransactionType. Pour plus d'informations sur la méthode setTransactionType, reportez-vous à la section concernant le suivi des performances d'ObjectGrid par le biais de l'infrastructure d'analyse des performances (PMI) de WebSphere Application Server.

Exécution de la méthode LogSequence

WebSphere eXtreme Scale peut propager des ensembles de modifications de mappe en programmes d'écoute ObjectGrid pour répartir les mappes d'une machine virtuelle Java™ à une autre. Pour que le programme d'écoute puisse traiter les LogSequences plus facilement, l'interface Session fournit la méthode processLogSequence. Cette méthode examine chaque LogElement dans l'objet LogSequence et effectue l'opération appropriée, par exemple une insertion, une mise à jour, une invalidation, etc. sur la BackingMap identifiée par LogSequence MapName. Une session ObjectGrid doit être disponible avant l'appel de la méthode processLogSequence. L'application a également la responsabilité d'effectuer les appels de validation ou d'annulation appropriés pour terminer la session. L'auto-validation n'est pas disponible pour cet appel de méthode. Le traitement normal par l'ObjectGridEventListener récepteur au niveau de la JVM distante est de démarrer une session en utilisant la méthode beginNoWriteThrough, qui empêche la propagation sans fin des modifications , puis d'appeler la méthode processLogSequence, et enfin de valider et d'annuler la transaction.

```
// Utilisez l'objet Session transmis au cours de
//ObjectGridEventListener.initialization...
session.beginNoWriteThrough();
// traitez la LogSequence reçue
try {
        session.processLogSequence(receivedLogSequence);
} catch (Exception e) {
        session.rollback(); throw e;
}
// validez les modifications
session.commit();
```

Méthode markRollbackOnly

Cette méthode est utilisée pour marquer la transaction actuelle en tant que "rollback only" (annulation uniquement). En marquant la transaction "rollback only", vous vous assurez que, même si la méthode de validation est appelée par une application, la transaction est annulée. Cette méthode est généralement utilisée par ObjectGrid lui-même ou par l'application lorsqu'une corruption de données est susceptible de se produire si la transaction est validée. Une fois cette méthode appelée, l'objet Throwable transmis à cette méthode est chaîné à l'exception com.ibm.websphere.objectgrid.TransactionException qui résulte de la méthode de validation si elle est appelée sous une session précédemment marquée comme "rollback only". Tout appel ultérieur de cette méthode pour une transaction déjà marquée en tant que "rollback only" est ignoré. Cela signifie que seul le premier appel transmettant une référence Throwable non nul est utilisé. Une fois la transaction marquée terminée, la marque "rollback only" est supprimée afin que la prochaine transaction lancée par la session soit validée.

Méthode isMarkedRollbackOnly

Renvoie un résultat si Session est marquée "rollback only". Cette méthode renvoie la valeur booléenne True si et uniquement si la méthode markRollbackOnly a été précédemment appelée sur cette Session et si la transaction commencée par cette Session est toujours active.

Méthode setTransactionTimeout

Définissez le délai d'attente de la prochaine transaction démarrée par cette Session sur un nombre spécifique de secondes. Cette méthode n'affecte pas le délai d'attente des transactions précédemment commencées par cette Session. Cela affecte uniquement les transactions lancées après l'appel de la méthode. Si cette méthode n'est jamais appelée, la valeur de délai d'attente passée à la méthode setTxTimeout de la méthode com.ibm.websphere.objectgrid.ObjectGrid est utilisée.

Méthode getTransactionTimeout

Cette méthode renvoie la valeur de délai d'attente de la transaction, exprimée en secondes. La dernière valeur passée en tant que valeur de délai d'attente à la méthode setTransactionTimeout est renvoyée par cette méthode. Si la méthode setTransactionTimeout n'est jamais appelée, la valeur de délai d'attente passée à la méthode setTxTimeout de la méthode com.ibm.websphere.objectgrid.ObjectGrid est utilisée.

transactionTimedOut

Cette méthode renvoie une valeur booléenne si le délai d'attente de la transaction actuelle commencée par cette Session a été dépassé.

Méthode isFlushing

La méthode renvoie la valeur booléenne True si et uniquement si toutes les modifications de transaction sont vidées vers le plug-in Loader comme conséquence de la méthode de vidage de l'interface Session appelée. Cette méthode peut être utile à un plug-in Loader lorsqu'il doit savoir pourquoi la méthode batchUpdate a été appelée.

Méthode isCommitting

Cette méthode renvoie la valeur booléenne True si et uniquement si toutes les modifications de transaction sont validées comme conséquence de la méthode de validation de l'interface de Session appelée. Cette méthode peut être utile à un plug-in Loader lorsqu'il doit savoir pourquoi la méthode batchUpdate a été appelée.

Méthode setRequestRetryTimeout

Cette méthode définit, en millisecondes, la valeur de délai d'attente avant la prochaine tentative de requête pour la session. Si le client définit une valeur de délai d'attente avant la prochaine tentative de requête, le paramètre de la session remplace la valeur du client.

Méthode getRequestRetryTimeout

Cette méthode récupère le paramètre de délai d'attente avant la prochaine tentative de requête sur la session. La valeur `-1` indique que le délai d'attente n'est pas défini. La valeur `0` indique qu'il est en mode d'interruption immédiate. Une valeur supérieure à `0` indique le paramètre de délai d'attente, en millisecondes.

Rubrique parent : Plug-in d'indexation des données

Java

Configuration du plug-in HashIndex

Vous pouvez configurer le plug-in HashIndex intégré (classe com.ibm.websphere.objectgrid.plugins.index.HashIndex) à l'aide d'un programme, ou à l'aide d'un index dynamique.

Pourquoi et quand exécuter cette tâche

La configuration d'un index composite s'effectue de la même manière que celle d'un index standard avec XML, à l'exception de la valeur de la propriété **attributeName**. Dans un index composite, la valeur de la propriété **attributeName** est une liste d'attributs séparés par une virgule. Par exemple, la classe de valeur Address a trois attributs : city, state et zipcode. Un index composite peut être défini avec la valeur de la propriété **attributeName** `"city,state,zipcode"` pour indiquer que la ville, l'état et le code postal sont inclus dans l'index composite.

Notez également qu'un index HashIndex composite ne prend pas en charge les recherches de plages et que sa propriété RangeIndex ne peut pas prendre la valeur true.

Procédure

Configuration d'un index composite à l'aide d'un programme. Ne s'applique qu'à un index dynamique.

L'exemple de code suivant crée le même index composite :

```
          HashIndex mapIndex = new HashIndex();
    mapIndex.setName("Address.CityStateZip");
    mapIndex.setAttributeName(("city,state,zipcode"));
    mapIndex.setRangeIndex(true);



BackingMap bm = objectGrid.getMap(“mymap”);
        bm.createDynamicIndex(mapIndex, null);

        try {
            // Si DynamicIndexCallback n'est pas utilisé, attendre que l'index soit prêt.
            // Le délai d'attente dépend de la taille actuelle de la mappe
            Thread.sleep(3000);
        } catch (Throwable t) {
            // ...
        }

        // Lorsque l'index est prêt, les applications peuvent tenter d'obtenir l'instance
d'interface
        // de l'index d'application.
        // Les applications doivent trouver un moyen de vérifier que l'index est prêt à
être utilisé,
        // faute de quoi elles utilisent l'interface DynamicIndexCallback.
        // L'exemple ci-dessous illustre comment attendre que l'index soit prêt
        // Prenez en compte la taille de la mappe dans le délai d'attente total.

        Session session = obectGrid.getSession();
        ObjectMap m = session.getMap("mymap");
        MapRangeIndex codeIndex = null;

        int counter = 0;
        int maxCounter = 10;
        boolean ready = false;
        while (!ready && counter < maxCounter) {
            try {
                counter++;
                codeIndex = (MapRangeIndex) m.getIndex("Address.CityStateZip");
                ready = true;
            } catch (IndexNotReadyException e) {
                // implique que l'index n'est pas prêt, ...
                System.out.println("Index non prêt. Veuillez patienter.");
                try {
                    Thread.sleep(3000);
                } catch (Throwable tt) {
```

```
                    // ...
                }
            } catch (Throwable t) {
                // exception inattendue
                t.printStackTrace();
            }
        }

        if (!ready) {
            System.out.println("Index non prêt. Remédier à cette situation.");
        }
```

Attributs du plug-in HashIndex

Vous pouvez utiliser les attributs suivants pour configurer le plug-in HashIndex

Utilisation d'un index composite

L'index HashIndex composite permet d'améliorer les performances des requêtes et d'éviter des recherches dans les mappes, consommatrices en ressources. Il facilite également les recherches d'objets mis en cache effectuées par l'API HashIndex lorsque les critères de recherche contiennent plusieurs attributs.

2.5+ Utilisation d'un index global

La mise en oeuvre d'un index global peut améliorer les performances de la recherche de données dans un grand environnement partitionné, comportant, par exemple 100 partitions.

Rubrique parent : Plug-in d'indexation des données

Java

Attributs du plug-in HashIndex

Vous pouvez utiliser les attributs suivants pour configurer le plug-in HashIndex

Attributs

Name

Spécifie le nom de l'index. Le nom doit être unique pour chaque mappe. Ce nom est utilisé pour extraire l'objet d'index de l'instance de mappe d'objet pour la mappe de sauvegarde.

AttributeName

Spécifie à l'index les noms des attributs, séparés par des virgules. Pour les index d'accès par zone, les noms d'attributs sont équivalents aux noms des zones. Pour les index d'accès par propriété, les noms d'attributs sont les noms de propriétés compatibles JavaBean. Si un seul nom d'attribut existe, HashIndex est un index d'attribut unique. Si cet attribut est une relation, il est également un index de relation. Si les noms d'attributs recouvrent plusieurs attributs, l'index HashIndex sera un index composite.

FieldAccessAttribute

Utilisé pour les mappes de non-entité. Si la valeur est égale à `true`, l'accès à l'objet s'effectue directement par les zones. S'il n'est pas défini ou a la valeur `false`, la méthode getter de l'attribut est utilisée pour accéder aux données.

GlobalIndexEnabled

Si la valeur est true, l'index global est activé et l'application peut transtyper l'objet d'index extrait en interface MapGlobalIndex.

Lorsque la propriété GlobalIndexEnabled de HashIndex a la valeur " true", la fonction d'index global de HashIndex est activée pour prendre en charge l'interface MapGlobalIndex sur une configuration HashIndex. Elle fournit un moyen efficace de rechercher des données dans un grand environnement partitionné.

POJOKeyIndex

Utilisé pour les mappes de non-entité. Si `true`, l'index introspecte l'objet dans la partie clé de la mappe. Ce paramètre est utile lorsque la clé est une clé composite et que la valeur ne contient pas la clé intégrée. Si l'attribut n'est pas défini ou que sa valeur est `false`, l'index introspecte l'objet dans la partie clé de la mappe.

RangeIndex

Si `true`, l'indexation de plage est activée et l'application peut transtyper l'objet d'index extrait vers l'interface MapRangeIndex. Si la propriété **`RangeIndex`** a la valeur `false`, l'application ne peut transtyper l'objet d'index extrait que vers l'interface MapIndex.

HashIndex à attribut unique ou HashIndex composite ?

Lorsque la propriété **`AttributeName`** de l'index HashIndex contient plusieurs noms d'attribut, l'index HashIndex est un index composite. Dans le cas contraire, si l'index inclut un seul nom d'attribut, il s'agit d'un index à attribut unique. Par exemple, la valeur de la propriété AttributeName d'un HashIndex composite pourrait être `city,state,zipcode`. Dans ce cas, l'index contient trois attributs séparés par des virgules. Si la valeur de la propriété **`AttributeName`** est uniquement `zipcode`, qui n'a qu'un seul attribut, il s'agit d'un index HashIndex à attribut unique.

Les index HashIndex composites constituent un moyen efficace pour consulter des objets mis en cache lorsque les critères de recherche impliquent plusieurs attributs. Toutefois, ils ne prennent pas en charge les index de plage et la propriété RangeIndex doit avoir la valeur `false`.

Pour plus d'informations, voir Utilisation d'un index composite.

HashIndex de relation

Si l'attribut indexé d'un HashIndex à attribut unique est une relation, que ce soit à valeur unique ou à valeurs multiples, l'index HashIndex est un HashIndex de relation. Pour l'index HashIndex de relation, la propriété RangeIndex de l'index HashIndex doit être définie avec la valeur false.

HashIndex de clés

Dans le cas de mappes de non-entité, lorsque la propriété **`POJOKeyIndex`** de HashIndex a la valeur `true`, l'index HashIndex est un index HashIndex de clés et la partie clé de l'entrée est utilisée pour l'indexation. Lorsque la propriété AttributeName du HashIndex n'est pas spécifiée, la totalité de la clé est indexée. Sinon, le HashIndex de clés ne peut être qu'un HashIndex à attribut unique.

HashIndex de plage

Lorsque la propriété RangeIndex de HashIndex a la valeur `true`, l'index HashIndex est un index de plage et il peut prendre en charge l'interface MapRangeIndex. Une implémentation MapRangeIndex prend en charge des fonctions de recherche de données à l'aide des fonctions de plage telles que supérieur à, inférieur à ou les deux, alors qu'un index MapIndex ne prend en charge que les fonctions d'égalité (equals). Pour un index à attribut unique, la propriété **RangeIndex** ne peut avoir la valeur `true` que si l'attribut indexé est de type Comparable. Si l'index à attribut unique est utilisé par la requête, la propriété RangeIndex doit avoir la valeur `true` et l'attribut indexé doit être de type Comparable. Pour l'index HashIndex de relation et l'index HashIndex composite, la propriété RangeIndex doit avoir la valeur `false`.

L'exemple qui précède est un index HashIndex de plage car la propriété RangeIndex a la valeur `true`.

Le tableau qui suit récapitule l'utilisation de l'index de plage.

Tableau 1. Prise en charge de l'index de plage. Indique si les types de HashIndex prennent ou non en charge l'index de plage.

Type de HashIndex	Prise en charge de l'index de plage
HashIndex à attribut unique : la clé ou l'attribut indexé est de type Comparable	Oui
HashIndex à attribut unique : la clé ou l'attribut indexé n'est pas de type Comparable	Non
HashIndex composite	Non
HashIndex de relation	Non

Rubrique parent : Configuration du plug-in HashIndex

Java

Utilisation d'un index composite

L'index HashIndex composite permet d'améliorer les performances des requêtes et d'éviter des recherches dans les mappes, consommatrices en ressources. Il facilite également les recherches d'objets mis en cache effectuées par l'API HashIndex lorsque les critères de recherche contiennent plusieurs attributs.

Amélioration des performances

Un index HashIndex composite constitue un outil rapide et pratique pour rechercher des objets mis en cache lorsque plusieurs attributs sont indiqués dans les critères de recherche. Il prend en charge les recherches de correspondance d'attribut complète, mais pas les recherches de plages.

Remarque : Les index composites ne prennent pas en charge l'opérateur BETWEEN dans le langage de requête ObjectGrid car la prise en charge des plages est obligatoire pour cet opérateur. De même, les opérateurs conditionnels "supérieur à" (>) et "inférieur à" (<) ne fonctionnent pas, pour la même raison.

Configuration d'un index composite

Vous pouvez configurer une indexation composite à l'aide d'un programme en utilisant un index dynamique.

Configuration par programmation

L'exemple suivant crée un index composite.

```
        HashIndex mapIndex = new HashIndex();
    mapIndex.setName("Address.CityStateZip");
    mapIndex.setAttributeName(("city,state,zipcode"));
    mapIndex.setRangeIndex(false);

  BackingMap bm = objectGrid.getMap("mymap");
    bm.createDynamicIndex(mapIndex, null);
```

Notez que la configuration d'un index composite est identique à la configuration d'un index standard avec XML, à l'exception de la valeur de la propriété attributeName. Dans un index composite, cette valeur est une liste d'attributs séparés par des virgules. Par exemple, la classe de valeur Address a trois attributs : city, state et zipcode. Un index composite peut être défini avec la valeur de propriété attributeName `"city,state,zipcode"` indiquant que la ville, l'état et le code postal sont inclus dans l'index composite.

Les index HashIndexes composites ne prennent pas en charge les recherches de plages et la propriété RangeIndex ne peut donc pas avoir la valeur true.

Exécution de recherches avec un index composite

Une fois un index composite configuré, une application peut utiliser la méthode findAll(Object) de l'interface MapIndex pour exécuter des recherches.

2.5+ **Restriction :** MapIndex.EMPTY_VALUE n'est pas pris en charge pour les index globaux composites.

```
Session sess = objectgrid.getSession();
ObjectMap map = sess.getMap("MAP_NAME");
MapIndex codeIndex = (MapIndex) map.getIndex("INDEX_NAME");
Object[] compositeValue = new Object[]{ MapIndex.EMPTY_VALUE,
                        "MN", "55901"};
Iterator iter = mapIndex.findAll(compositeValue);
// Fermer la session (facultatif dans les versions 7.1.1 et ultérieures) pour améliorer
les performances
sess.close();
```

La valeur MapIndex.EMPTY_VALUE est affectée à compositeValue[0] qui indique que l'attribut city est exclu de l'évaluation. Seuls les objets dont l'attribut state est égal à `"MN"` et l'attribut zipcode est égal à `"55901"` figureront dans le résultats.

Migration et interopérabilité

La seule contrainte liée à l'utilisation d'un index composite est qu'une application ne peut pas le configurer dans un environnement réparti contenant des conteneurs hétérogènes. Les anciens et nouveaux serveurs ne peuvent pas être combinés, car les anciens serveurs de conteneur ne reconnaissent pas les configurations d'index composite. L'index composite est semblable à un index d'attribut normal, mais il permet en outre l'indexation de plusieurs attributs. En cas d'utilisation d'un index d'attribut standard, un environnement

composé de plusieurs types de conteneur est viable.

Rubrique parent : Configuration du plug-in HashIndex

Utilisation d'un index global

2.5+ La mise en oeuvre d'un index global peut améliorer les performances de la recherche de données dans un grand environnement partitionné, comportant, par exemple 100 partitions.

Cette fonction fournit également un moyen de rechercher l'emplacement des attributs indexés et peut améliorer les opérations des agents ou des requêtes qui sont associées aux attributs indexés. Reportez-vous à la documentation de l'API MapGlobalIndex pour plus de détails sur les fonctions de l'index global.

Amélioration des performances

Dans un grand environnement partitionné, les objets mis en cache sont répartis dans toutes les partitions. Par conséquent, toute recherche de données utilisant un index, des requêtes ou des agents devra être exécutée sur tous les serveurs pour obtenir des résultats complets. Ce type de recherche est donc lent en raison des appels distants requis pour charger et explorer chaque partition. En outre, toutes les partitions ne contiennent pas des données qui correspondent aux critères de recherche. Un index global améliore les performances de recherche car il n'effectue les recherches que dans les partitions qui contiennent des données qui correspondent. La fonction d'index global peut suivre l'emplacement des attributs indexés et déterminer les partitions applicables pour les attributs parmi toutes les partitions. Généralement, les partitions applicables sont un sous-ensemble de toutes les partitions. Par conséquent, l'exécution des index, des requêtes et des agents sur les partitions applicables est beaucoup plus rapide que leur exécution sur toutes les partitions, même lorsque cela est compensé par l'index global.

Recherche de données

Les applications peuvent rechercher des données en utilisant des clés. Elles peuvent aussi utiliser des index si les données comportent un ou plusieurs attributs pour lesquels des index sont définis. Traditionnellement, les applications peuvent utiliser un proxy d'index client pour obtenir les clés d'entrée de toutes les partitions, ou utiliser un agent pour effectuer une recherche d'index sur toutes les partitions et renvoyer des clés de cache, des valeurs ou les deux. Grâce à la fonction d'index global, les applications peuvent rechercher des clés d'entrée, des valeurs ou les deux via l'API MapGlobalIndex, selon une approche efficace qui n'exécute les opérations que sur les partitions applicables.

Fonctionnement de l'agent

Si une opération d'agent est lié à des attributs indexés (par exemple, invalidation des entrées à l'aide d'attributs indexés), les applications peuvent utiliser l'index global pour rechercher en premier lieu les partitions applicables d'après les attributs. Ensuite, l'application peut envoyer l'agent à ces partitions applicables. Utilisez la méthode MapGlobalIndex.findPartitions() pour rechercher les partitions applicables en utilisant des attributs.

Activation d'un index global

L'index global est une extension du plug-in HashIndex, qui peut être activée sur n'importe quelle configuration HashIndex existante.

Exécution d'une recherche d'index globale

La fonction d'index global est défini dans l'API MapGlobalIndex. Après l'activation de l'index global dans un plug-in HashIndex, l'application peut transtyper un proxy d'index obtenu en type MapGlobalIndex et commencer à l'utiliser.

```
// in client ObjectGrid process
MapGlobalIndex mapGlobalIndexCODE = (MapGlobalIndex)m.getIndex("CODE", false);
Object[] attributes = new Object[] {new Integer(1)};
Collection partitions = mapGlobalIndexCODE.findPartitions(attributes);
Set keys = mapGlobalIndexDependency.findKeys(attributes);
Set values = mapGlobalIndexDependency.findValues(attributes);
Map entries = mapGlobalIndexDependency.findEntries(attributes);
```

Migration et interopérabilité

La seule contrainte liée à l'utilisation de l'index global réside dans le fait qu'une application ne peut pas le configurer dans un environnement réparti contenant des conteneurs hétérogènes. Les anciens et nouveaux serveurs ne peuvent pas être combinés, car les anciens serveurs de conteneur ne reconnaissent pas les configurations avec index global ou avec index global composite.

Pour utiliser un index global ou un index global composite, vous devez arrêter préalablement tous les serveurs de conteneur et les clients d'une application. Ensuite, activez l'index global sur la configuration HashIndex et redémarrez les serveurs de conteneur et les clients.

Rubrique parent : Configuration du plug-in HashIndex

Notification aux clients des mises à jour de mappe en utilisant l'interrogation continue

2.5+ Vous pouvez être notifié dans votre machine JVM (Java™ virtual machine) client de l'insertion ou de la mise à jour d'objets ou d'entrées dans la grille de données.

Avant de commencer

Si vous souhaitez utiliser l'interrogation continue, vous devez activer IBM® eXtremeIO, un mécanisme de transport utilisé pour communiquer entre les serveurs et les clients. Pour plus d'informations sur l'activation d'eXtremeIO, voir Configuration d'IBM eXtremeIO (XIO).

Pourquoi et quand exécuter cette tâche

Lorsque vous développez des applications client qui interagissent avec la grille de données, vous pouvez avoir besoin de requêtes qui extraient automatiquement des résultats en temps réel lorsque de nouvelles entrées qui correspondent aux critères de filtrage sont insérées ou mises à jour. Par exemple, vous développez une application de cotation boursière qui nécessite des mises à jour fréquentes. Ces mises à jour correspondent aux fluctuations des cours. Par conséquent, il est essentiel que l'application soit informée des changements immédiatement afin de fournir des résultats précis et en temps opportun. L'interrogation continue, qui a un faible impact sur la mémoire, peut informer de manière proactive les clients des modifications qui se produisent dans la grille de données.

Procédez comme suit pour programmer les applications client pour qu'elles utilisent l'interrogation continue.

Restriction : Les requêtes qui spécifient un chemin d'attribut de valeur `null` ne sont pas prises en charge si l'objet de valeur n'est pas un type primitif Java, tel qu'une chaîne ou un entier. Lorsque `null` est spécifié,le filtre de requête est utilisé pour interroger l'objet de valeur tout entier.

Procédure

1. Appelez le gestionnaire d'interrogation continue dans l'application client. Par exemple, insérez la ligne de code suivante :

   ```
   ContinuousQueryManager cqMan = ContinuousQueryManagerFactory.getManager(og);
   ```

2. Définissez un filtre ou une chaîne de filtres. Vous pouvez implémenter vos propres filtres ou les filtres de base `AND, OR, LT, GT, EQ`, etc. qui sont fournis. Des identifiants uniques sont affectés aux filtres ou aux chaînes de filtres. Pour plus d'informations sur les filtres pris en charge, voir Accès à la documentation des API Java pour rechercher les API d'interrogation continue.

   L'exemple de code suivant montre une manière d'utiliser le filtre de base equals (EQ). Supposons que la grille de données contienne des objets Customer avec la zone `firstName` et que le filtre renvoie `true` lorsque `firstName` est égal à `Larry`.

   ```
   EQFilter<String, String> equalsFilter = new EQFilter<String, String>("firstName",
   "Larry");
   ```

3. Définissez une requête en utilisant le filtre que vous avez créé au cours de l'étape précédente ; par exemple :

   ```
   ContinuousQueryTopic<String, Customer> topic =
     cqMan.<String, Customer> defineContinuousQuery("myMapName", equalsFilter, true,
   true, true);
   ```

4. Facultatif : Indiquez que le cache de requête doit accéder aux résultats côté client de l'interrogation continue. Si la requête est définie comme requête de clés, seules les clés qui correspondent à la requête se trouvent dans le cache d'interrogation continue. Par exemple :

   ```
   ContinuousQueryCache cache = topic.getCache();
   ```

5. Facultatif : En outre, vous pouvez enregistrer une classe qui implémente l'interface ContinuousQueryListener avec une instance ContinuousQueryTopic pour recevoir la modification de l'interrogation continue. Appelez la méthode addListener pour enregistrer le programme d'écoute. Par exemple :

   ```
   ContinuousQueryListener<String, Customer> listener = new MyCQListener<String,
   Customer>();
   topic.addListener(listener);
   ```

Que faire ensuite

Voir Documentation des API : Package com.ibm.websphere.objectgrid.continuousquery pour plus d'informations sur l'API d'interrogation continue.

Rubrique parent : Développement d'applications de grilles de données avec des API Java

Développement d'applications de grille de données avec la passerelle REST

Vous pouvez utiliser la passerelle REST (Representational State Transfer) pour accéder à des grilles de données simples qui sont hébergées par une collectivité. Cette passerelle REST est pratique lorsque vous devez accéder à une grille de données à partir d'environnements non Java.

Avant de commencer

- Vous pouvez utiliser la passerelle REST avec WebSphere DataPower XC10 Appliance version 8.6 ou une version ultérieure.
- Vous devez créer une grille de données simple sur le dispositif. Pour plus d'informations sur la création d'une grille de données simple, voir Création de grilles de données simples.

Pourquoi et quand exécuter cette tâche

Utilisez la passerelle REST pour accéder aux données d'une grille de données simple à partir d'environnements non Java, tels que le dispositif DataPower XI50 ou une application .NET. Vous pouvez aussi utiliser la passerelle REST pour accéder aux données d'une mappe à partir d'une machine virtuelle Java™ qui ne peut pas héberger la fonction ORB IBM® utilisée par l'API ObjectMap basée sur Java.

Transactions

Chaque opération REST sur WebSphere DataPower XC10 Appliance commence et termine une transaction indépendante sur la grille de données. Il n'est pas possible de chaîner ensemble plusieurs opérations dans une même transaction.

Equilibrage de charge

Lorsque vous utilisez la passerelle REST, il incombe aux clients d'équilibrer la charge de leurs requêtes sur la collectivité WebSphere DataPower XC10 Appliance. Vous pouvez utiliser un équilibreur de charge externe ou ajouter de la logique supplémentaire dans le client HTTP que vous utilisez dans le programme client.

Sécurité

Communiquer via la passerelle REST permet toujours de disposer d'une configuration sécurisée, même si la sécurité n'est pas activée sur la grille de données. Configurez les groupes d'utilisateurs qui doivent accéder à la grille de données avec `tous` les droits d'accès à cette grille.

Relation avec le service de données REST WebSphere eXtreme Scale

La passerelle REST est une entité distincte du service de données REST WebSphere eXtreme Scale, qui implémente l'interface de services de données Microsoft ADO.NET.

2.5+ Alias de grille

Lorsque vous devez remplir simultanément plusieurs grilles de données, vous pouvez utiliser la passerelle REST pour créer et gérer un alias de grille. Un alias de grille vous permet de passer d'une grille à une autre et de les remplir simultanément. Par exemple, supposons que vous vouliez effectuer des opérations sur la grille Grille A et que vous vouliez aussi une couche d'adressage indirect pour pouvoir basculer vers une autre grille, Grille B, afin de la remplir. Vous pouvez créer un alias, Grille C, qui pointe vers Grille A. Vous pouvez alors utiliser cet alias pour effectuer des opérations sur Grille A tout en remplissant Grille B. Pendant que vous effectuez les opérations sur Grille A (à l'aide de l'alias Grille C), vous pouvez effectuer une opération REST pour que cet alias pointe vers Grille B. Les alias de grille résident dans la ressource REST `resources/gridalias`, dans laquelle ils peuvent être créés, interrogés et supprimés. Pour plus d'informations concernant la gestion des alias de grille à l'aide de la passerelle REST, voir Passerelle REST : Opérations REST.

Passerelle REST : Format d'URI
En spécifiant un URI dans un format spécifique, vous pouvez accéder à votre grille de données simple et y effectuer des opérations.

Passerelle REST : Format des données
La passerelle REST utilise l'en-tête Content-Type de vos requêtes HTTP pour déterminer le format des données stockées dans la grille de données.

Passerelle REST : Opérations REST
Vous utilisez des opérations HTTP POST, GET et DELETE pour insérer, mettre à jour, obtenir et supprimer des données dans la grille de données. La passerelle REST prend également en charge les requêtes HTTP de gestion d'un alias de grille pointant vers votre grille de données. Les alias de grille sont utiles lorsque vous devez remplir plusieurs grilles simultanément et que vous devez passer de

l'une à l'autre. Il est possible de créer, interroger et supprimer un alias de grille. Ces alias utilisent la ressource REST `/resource/gridalias`.

Exemple de passerelle REST : Insertion et obtention d'entrées de mappe de grille de données
Vous pouvez utiliser les méthodes HTTP POST et GET pour insérer et obtenir des entrées de mappe de grille de données.

Exemple utilisant une passerelle REST : Insertion de données dans une mappe REST et accès à ces données, à partir d'un client Java et à l'aide des API ObjectMap
Lorsque des données sont insérées dans une mappe à l'aide de la passerelle REST, une classe d'encapsuleur de type com.ibm.websphere.xsa.RestValue est utilisée pour encapsuler le type de contenu (content-type) et le corps (body) de requête fournis. Vous pouvez utiliser la même classe RestValue pour insérer des données dans la mappe et obtenir des données de celle-ci à partir d'un client Java et en utilisant les API ObjectMap.

Exemple de passerelle REST : Effacement d'entrées de mappe de grille de données
Vous pouvez utiliser la méthode HTTP DELETE de la passerelle REST pour effacer une mappe dans une grille de données.

Exemple de passerelle REST : Création de mappes dynamiques
Lorsque vous créez une grille de données simple, une mappe par défaut de même nom est créée par défaut. Vous pouvez aussi utiliser des modèles de mappes pour créer d'autres mappes selon les besoins de votre application.

Exemple de passerelle REST : Expiration de la durée de vie (TTL, Time to live)
Vous pouvez définir une valeur de durée de vie pour les mappes last update time (*.LUT) et last access time (*.LAT). La valeur de durée de vie (TTL) par défaut pour les deux types de mappes est d'une heure.

Passerelle REST : Configuration de la sécurité
Pour accéder à une grille de données via la passerelle REST, l'utilisateur doit être authentifié sur WebSphere DataPower XC10 Appliance, que la sécurité de la grille de données ait été activée ou non. Le client d'application doit toujours fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP. Pour accéder à des grilles de données via la passerelle REST, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation.

Passerelle REST : sessions HTTP et cookies
Utilisez des sessions HTTP et des cookies avec la passerelle REST grâce aux en-têtes `Set-Cookie:`.

Rubrique parent : Développement d'applications pour accéder à des grilles de données simples

Tâches associées:
Création de grilles de données simples

Référence associée:
Fichier de propriétés du client
Options de configuration de mappe dynamique

Passerelle REST : Format d'URI

En spécifiant un URI dans un format spécifique, vous pouvez accéder à votre grille de données simple et y effectuer des opérations.

Format d'URI

L'URI REST pour accéder à une grille de données simple sur WebSphere DataPower XC10 Appliance a le format suivant :

```
/resources/datacaches/[nom_grille]/[nom_mappe]/[clé]
```

La racine de contexte par défaut est `resources`.

Si vous créez une grille de données simple nommée `MyDataGrid` sur le dispositif avec le nom d'hôte `myxc10.ibm.com`, l'URL résultante pour accéder au nom de clé `my.data.item` sera :

```
http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyMap/my.data.item
```

Dans l'exemple précédent, la mappe par défaut `MyMap` est utilisée dans la grille `MyDataGrid`. Cette mappe par défaut n'a pas d'éviction de durée de vie. Les entrées placées dans la grille de données y restent tant qu'elles ne sont pas explicitement supprimées. Pour configurer l'éviction de durée de vie, voir Exemple de passerelle REST : Expiration de la durée de vie (TTL, Time to live).

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Passerelle REST : Format des données

La passerelle REST utilise l'en-tête Content-Type de vos requêtes HTTP pour déterminer le format des données stockées dans la grille de données.

Format des données

La passerelle REST utilise l'en-tête Content-type de vos requêtes HTTP pour déterminer le format des données stockées dans la grille de données. Si vous insérez du contenu de type `application/xml`, lorsque votre application effectue une opération GET pour la même clé du cache, le corps de la réponse et le Content-type sont dans un type de format équivalent. Dans cet exemple, le corps de réponse sera au format `application/xml`. Vous pouvez stocker des données de plusieurs types de contenu dans la même grille de données. Voici des exemples de quelques types de contenu valides :

Tableau 1. Types de contenu pour l'en-tête content-type dans des requêtes HTTP

Type de contenu	Utilisez
`application/xml`	XML
`application/json`	Données JavaScript
`application/octet-stream`	Objets sérialisés, données de portée générale

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Passerelle REST : Opérations REST

Vous utilisez des opérations HTTP POST, GET et DELETE pour insérer, mettre à jour, obtenir et supprimer des données dans la grille de données. La passerelle REST prend également en charge les requêtes HTTP de gestion d'un alias de grille pointant vers votre grille de données. Les alias de grille sont utiles lorsque vous devez remplir plusieurs grilles simultanément et que vous devez passer de l'une à l'autre. Il est possible de créer, interroger et supprimer un alias de grille. Ces alias utilisent la ressource REST `/resource/gridalias`.

Opérations REST de remplissage de grilles de données

Tableau 1. Liste des opérations, accompagnées des méthodes HTTP équivalentes et des définitions des codes de réponse

Opération	Méthode HTTP	Code de réponse
Insérer ou mettre à jour	POST	• `200 CREATED` : Les données ont été insérées ou mises à jour correctement dans la grille de données. • `400 BAD REQUEST` : L'opération d'insertion ou de mise à jour des données ne s'est pas terminée correctement.
Obtenir	GET	• `200 OK` : Le corps et le type de contenu de la réponse sont extraits d'une opération d'insertion ou de mise à jour antérieure. • `404 NOT FOUND` : La clé spécifiée n'est pas présente dans la grille de données. • `400 BAD REQUEST` : Le dispositif n'a pas pu traiter la demande.
Supprimer	DELETE	• `200 NO CONTENT` : L'entrée a été supprimée de la grille de données. • `400 BAD REQUEST` : Le dispositif n'a pas pu traiter la demande.

2.5+

Opérations REST de gestion d'un alias de grille de données

Tableau 2. Liste des opérations, accompagnées des méthodes HTTP équivalentes et des définitions des codes de réponse

Opération	Méthode HTTP	Code de réponse
Ajouter ou mettre à jour un alias	POST `/resources/gridalias/<nom_alias>?src=<nom_grille_source>`	• `204 SUCCESS` : L'alias de grille a été créé. • `401 SOURCE GRID DOES NOT EXIST` : L'alias n'a pas pu être créé car la grille de données vers laquelle il pointe n'existe pas. • `409 DATA GRID EXISTS` : L'alias n'a pas pu être créé car une grille de données portant ce nom existe déjà.
Obtenir la grille en cours pour un alias	GET `/resources/gridalias/<nom_alias>`	• `200 OK`
Obtenir la liste de tous les alias	GET `/resources/gridalias/`	• `200 OK`
Supprimer un alias	DELETE `/resources/gridalias/<nom_alias>`	• `204 SUCCESS` : L'alias a été supprimé. • `404 BAD REQUEST` : L'alias n'existe pas.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Insertion et obtention d'entrées de mappe de grille de données

Vous pouvez utiliser les méthodes HTTP POST et GET pour insérer et obtenir des entrées de mappe de grille de données.

Exemple : Opération d'insertion

A l'aide de l'URI et du format de données définis, vous pouvez insérer des informations dans la grille de données. L'exemple suivant insère une clé "bob" dans la grille MyGrid et la mappe MyGrid :

```
POST /resources/datacaches/MyGrid/MyGrid/bob
Content-type: application/xml
<mesdonnées>des données</mesdonnées>
```

Exemple : Opération d'obtention

Pour récupérer cette clé qui a été insérée dans l'exemple précédent, vous pouvez utiliser l'URI suivant :

```
GET /resources/datacaches/MyGrid/MyGrid/bob
```

Vous devez exécuter les opérations GET sur une clé individuelle. Vous ne pouvez pas récupérer toutes les entrées de la mappe.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple utilisant une passerelle REST : Insertion de données dans une mappe REST et accès à ces données, à partir d'un client Java et à l'aide des API ObjectMap

Lorsque des données sont insérées dans une mappe à l'aide de la passerelle REST, une classe d'encapsuleur de type com.ibm.websphere.xsa.RestValue est utilisée pour encapsuler le type de contenu (content-type) et le corps (body) de requête fournis. Vous pouvez utiliser la même classe RestValue pour insérer des données dans la mappe et obtenir des données de celle-ci à partir d'un client Java et en utilisant les API ObjectMap.

Code du client Java permettant d'accéder aux mappes REST

```
RestValue rv = new RestValue();
rv.setContentType(“application/xml”);
String myXml=”<customer>brian</customer>”;
rv.setValue(myXml.getBytes(“UTF8”));
ogSession.begin();
ObjectMap map = ogSession.getMap(“myMap.LUT”);
map.insert(“brian”,rv);
ogSession.commit();
```

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Effacement d'entrées de mappe de grille de données

Vous pouvez utiliser la méthode HTTP DELETE de la passerelle REST pour effacer une mappe dans une grille de données.

Effacement d'une entrée individuelle

Pour supprimer une entrée individuelle, utilisez la méthode DELETE et le nom de clé de l'objet :

```
DELETE http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyDataGrid/my.data.item
```

Effacement d'une mappe entière sur la grille de données

Pour effacer une mappe entière dans la grille de données, utilisez la méthode HTTP DELETE et omettez la partie de l'URI qui concerne la clé. Par exemple, pour effacer la mappe MyDataMap.LUT sur la grille de données MyDataGrid, utilisez l'opération suivante :

```
DELETE http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyDataMap.LUT
```

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Création de mappes dynamiques

Lorsque vous créez une grille de données simple, une mappe par défaut de même nom est créée par défaut. Vous pouvez aussi utiliser des modèles de mappes pour créer d'autres mappes selon les besoins de votre application.

Création de mappe dynamique

La première opération sur une mappe qui correspond au modèle de mappe mais qui n'a pas encore été créée aboutit à la création d'une nouvelle mappe dynamique. Par exemple, pour créer une mappe dynamique en utilisant le modèle `*.LUT` `MyMap.LUT`, vous pouvez utiliser l'URI suivant dans une opération GET, DELETE ou POST :

```
http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyMap.LUT/a.key
```

Pour savoir comment nommer les mappes dynamiques, voir Options de configuration de mappe dynamique.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Exemple de passerelle REST : Expiration de la durée de vie (TTL, Time to live)

Vous pouvez définir une valeur de durée de vie pour les mappes last update time (*.LUT) et last access time (*.LAT). La valeur de durée de vie (TTL) par défaut pour les deux types de mappes est d'une heure.

Exemple

Pour définir une valeur de durée de vie pour les mappes de date/heure de dernière mise à jour (*.LUT) et de date/heure de dernier accès (*.LAT), spécifiez le paramètre de demande de duré de vie avec une valeur exprimée en secondes. Par exemple, pour définir une valeur de durée de vie de 600 secondes sur la clé `a.key`, spécifiez le paramètre de demande `ttl` lorsque la valeur est insérée ou mise à jour dans la grille de données à l'aide de la méthode HTTP POST :

```
http://myxc10.ibm.com/resources/datacaches/MyDataGrid/MyMap.LUT/a.key?ttl=600
```

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

Passerelle REST : Configuration de la sécurité

Pour accéder à une grille de données via la passerelle REST, l'utilisateur doit être authentifié sur WebSphere DataPower XC10 Appliance, que la sécurité de la grille de données ait été activée ou non. Le client d'application doit toujours fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP. Pour accéder à des grilles de données via la passerelle REST, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation.

Authentification et autorisation

Pour accéder à une mappe de grille de données via une passerelle REST, l'utilisateur ou le groupe d'utilisateurs doit être authentifié et autorisé à accéder à la grille de données spécifiée dans l'URI. Même si la sécurité n'est pas activée sur la grille de données, vous devez configurer le groupe d'utilisateurs que vous utilisez pour communiquer via la passerelle REST pour avoir un accès `intégral` à la grille de données. Pour plus d'informations sur la configuration de l'accès à la grille de données, voir Activation de la sécurité pour les grilles de données. Le client d'application doit fournir un en-tête d'autorisation de base avec l'ID et le mot de passe de l'utilisateur autorisé dans les en-têtes HTTP de la requête HTTP.

```
Authorization : Basic <chaîne codée en Base64 de "ID_utilisateur:mot_de_passe">
```

Pour plus d'informations sur le format de l'en-tête d'autorisation de base, voir Wikipédia : Authentification d'accès de base.

Grilles de données sécurisées

Vous pouvez utiliser la passerelle REST dans une configuration de grille de données sécurisée. Pour accéder aux grilles de données sécurisées, spécifiez l'ID utilisateur et le mot de passe dans un en-tête d'autorisation. L'utilisateur doit être authentifié et autorisé à accéder à la grille de données spécifiée dans l'URI.

Tableau 1. Grilles de données sécurisées

Permission	Get	Post	Supprimer
READ	X		
WRITE	X		
CREATE	X	X	
ALL	X	X	X

Sécurité du transfert

Les clients utilisant la passerelle REST peuvent utiliser le protocole HTTPS si la sécurité du transport est requise.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST
Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité
Droits utilisateur
Mot de passe xcadmin

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Passerelle REST : sessions HTTP et cookies

Utilisez des sessions HTTP et des cookies avec la passerelle REST grâce aux en-têtes `Set-Cookie:`.

La code de la passerelle REST crée une session HTTP lorsqu'elle reçoit une requête d'un client qui ne se trouve pas actuellement en session. Pour éviter la création inutile de sessions et conserver les meilleures performances, le client REST doit conserver les cookies qui sont renvoyés de la passerelle REST avec des en-têtes `Set-Cookie:` et fournir en retour ces mêmes cookies à la passerelle REST avec des en-têtes `Cookie:` lors des requêtes suivantes.

Rubrique parent : Développement d'applications de grille de données avec la passerelle REST

.NET

Développement d'applications de grille de données avec les API .NET

2.5+ Vous pouvez développer des applications Microsoft .NET qui utilisent la même grille de données que vos applications Java™.

.NET **Configuration de l'environnement de développement .NET**
Pour utiliser WebSphere eXtreme Scale Client for .NET dans Microsoft Visual Studio, vous devez installer l'environnement de développement et configurer le projet pour utiliser l'assemblage WebSphere eXtreme Scale Client for .NET.

.NET **Création de mappes dynamiques avec les API .NET**
Vous pouvez créer des mappes dynamiques avec des API .NET une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

.NET **Définition d'annotations ClassAlias et FieldAlias pour corréler des classes Java et .NET**
Utilisez des annotations ClassAlias et FieldAlias pour activer le partage des données de la grille de données entre les classes Java et .NET.

.NET **Mappage de clés avec des partitions avec des annotations PartitionKey**
Un alias PartitionKey est utilisé pour identifier les zones ou les attributs sur lesquels un calcul de code de hachage est exécuté pour déterminer la partition dans laquelle les données sont sauvegardées. L'annotation PartitionKey n'est valide que sur les attributs de clé.

.NET **Programmation de transactions dans des applications .NET**
Lorsque vous écrivez une application .NET qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

.NET **Configuration de la sécurité de grille de données pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer .NET et Java pour communiquer via SSL (Secure Sockets Layer) et utiliser la logique d'authentification UserPassword.

.NET **Configuration de TLS pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez configurer Transport Layer Security (TLS) pour WebSphere eXtreme Scale Client for .NET.

.NET **Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET**
Pour envoyer des données d'identification de WebSphere eXtreme Scale Client for .NET au serveur, vous devez implémenter les interfaces ICredentialGenerator et ICredential. Ces interfaces génèrent un objet de données d'identification qui est envoyé à la grille de données et interprété sur le serveur. Sur le serveur, cet objet est interprété par le plug-in correspondant.

.NET **Programmation de données d'identification personnalisées pour WebSphere eXtreme Scale Client for .NET**
Vous pouvez spécifier des données d'identification de l'utilisateur pour une mappe. De cette manière, deux utilisateurs peuvent interagir avec la même grille de données par l'intermédiaire d'une application Web.

Rubrique parent : Développement d'applications pour accéder à des grilles de données simples

.NET

Configuration de l'environnement de développement .NET

Pour utiliser WebSphere eXtreme Scale Client for .NET dans Microsoft Visual Studio, vous devez installer l'environnement de développement et configurer le projet pour utiliser l'assemblage WebSphere eXtreme Scale Client for .NET.

Avant de commencer

- Pour la liste des éditions Microsoft Visual Studio prises en charge, voir Remarques relatives à Microsoft .NET.
- Installez WebSphere eXtreme Scale Client for .NET. Dans l'assistant d'installation, choisissez le chemin **personnalisé** et sélectionnez l'environnement de développement. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client for .NET.

Procédure

1. Ouvrez le projet dans l'environnement Microsoft Visual Studio.
2. Ajoutez une référence à l'assemblage WebSphere eXtreme Scale Client for .NET. L'assemblage se trouve dans le répertoire *net_client_home*\bin. Choisissez le fichier `IBM.WebSphere.Caching.dll`.
3. Ajoutez les lignes suivantes à l'application pour utiliser les API WebSphere eXtreme Scale Client for .NET :

```
using IBM.WebSphere.Caching;
using IBM.WebSphere.Caching.Map;
```

Résultats

Lorsque vous intégrez les assemblages dans l'environnement de développement, IntelliSense est activé pour les API WebSphere eXtreme Scale Client for .NET.

Que faire ensuite

Utilisez WebSphere eXtreme Scale Client pour les API .NET dans l'application client. Pour plus d'information sur l'accès à la documentation des API, voir Accès à la documentation des API WebSphere eXtreme Scale Client for .NET.

.NET **Accès à la documentation des API WebSphere eXtreme Scale Client for .NET**
Vous pouvez accéder à la documentation des API WebSphere eXtreme Scale Client for .NET dans un fichier `.chm` ou en affichant cette documentation dans le centre de documentation.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

.NET

Accès à la documentation des API WebSphere eXtreme Scale Client for .NET

Vous pouvez accéder à la documentation des API WebSphere eXtreme Scale Client for .NET dans un fichier `.chm` ou en affichant cette documentation dans le centre de documentation.

Procédure

Utilisez les options suivantes pour ouvrir la documentation des API WebSphere eXtreme Scale Client for .NET :

- Utilisez la documentation des API .NET Client, installée avec le produit. Pour ouvrir la documentation des API client .NET localement, ouvrez le fichier *net_client_home*`\doc\IBM.WebSphere.Caching.chm`.
- Affichez la documentation des API dans le centre de documentation. Pour plus d'informations, voir la documentation relative au client pour l'API .NET.

Rubrique parent : .NET Configuration de l'environnement de développement .NET

.NET

Création de mappes dynamiques avec les API .NET

2.5+ Vous pouvez créer des mappes dynamiques avec des API .NET une fois la grille de données instanciée. Vous pouvez instancier de manière dynamique des mappes qui reposent sur un ensemble de modèles de mappe prédéfinis.

Avant de commencer

Choisissez les options de configuration que vous souhaitez utiliser dans votre mappe dynamique. Pour plus d'informations, voir Options de configuration de mappe dynamique.

Procédure

Appelez la méthode GetGridMapPessimisticTx.

```
IGridManager gm = GridManagerFactory.GetGridManager( );
ICatalogDomainInfo cdi =
        gm.CatalogDomainManager.CreateCatalogDomainInfo( catalogServerHostsList );
IClientConnectionContext ccc = gm.Connect( cdi, "SimpleClient.properties" );
grid = gm.GetGrid( ccc, "Grid" );
IGridMapPessimisticTx<Object, Object> map =
        grid.GetGridMapPessimisticTx<Object, Object>( "SessionState.LAT.P" );
```

La mappe SessionState.LAT.P une mappe qui utilise l'expulsion en fonction de l'heure de dernier accès, le verrouillage pessimiste et l'invalidation du cache local.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Référence associée:
Options de configuration de mappe dynamique

.NET

Programmation de transactions dans des applications .NET

Lorsque vous écrivez une application .NET qui nécessite des transactions, vous devez tenir compte, entre autres choses, de la gestion des verrous et des collisions et de l'isolement des transactions.

.NET **Interaction avec les données dans une transaction pour les applications .NET**
L'API de WebSphere eXtreme Scale Client impose que chaque unité d'exécution possède son propre objet IGridMapPessimisticTx ou IGridMapPessimisticAutoTx. Avec l'objet IGridMapPessimisticTx, la propriété Transaction sert à commencer, valider ou annuler explicitement la transaction. Avec l'objet IGridMapPessimisticAutoTx, les opérations de lancement, de validation et d'annulation de la transaction s'effectuent automatiquement. Utilisez des sessions pour interagir avec les données à l'aide des opérations Add, Put et Replace.

.NET **Configuration et mise en oeuvre du verrouillage dans les applications .NET**
Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste. Vous pouvez également remplacer le délai de verrouillage d'une instance de mappe. Une fois le verrouillage configuré, vous pouvez verrouiller des clés individuelles ou une liste de clés de la mappe.

.NET **Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET**
Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Concepts associés:
Transactions

.NET

Interaction avec les données dans une transaction pour les applications .NET

L'API de WebSphere eXtreme Scale Client impose que chaque unité d'exécution possède son propre objet IGridMapPessimisticTx ou IGridMapPessimisticAutoTx. Avec l'objet IGridMapPessimisticTx, la propriété Transaction sert à commencer, valider ou annuler explicitement la transaction. Avec l'objet IGridMapPessimisticAutoTx, les opérations de lancement, de validation et d'annulation de la transaction s'effectuent automatiquement. Utilisez des sessions pour interagir avec les données à l'aide des opérations Add, Put et Replace.

Pourquoi et quand exécuter cette tâche

Les interfaces IGridMapPessimisticTx et IGridMapPessimisticAutoTx fournissent des opérations telles que Add, Get, Put, Replace et Remove pour manipuler les données. L'interface IGridMapPessimisticTx offre des opérations supplémentaires, telles que Lock et GetAndLock, qui permettent de contrôler les accès simultanés aux données.

Procédure

- Ajout de données.

  La fragment de code suivant montre comment utiliser l'interface IGridMapPessimisticTx pour commencer une nouvelle transaction, créer un élément pour la grille de données, puis valider la transaction :

  ```
  IGridMapPessimisticTx<String,Person> ptmap;
  ptmap = grid.GetGridMapPessimisticTx<String,Person>("PERSON");
  ptmap.Transaction.Begin();
  Person p = new Person();
  p.name = "John Doe";
  ptmap.Add(p.name, p);
  ptmap.Transaction.Commit();
  ```

- Remplacement de données.

  La fragment de code suivant montre comment utiliser l'interface IGridMapPessimisticTx pour commencer une nouvelle transaction, verrouiller un élément dans la grille de données et obtenir sa valeur, remplacer la valeur de cet élément, puis valider la transaction :

  ```
  IGridMapPessimisticTx<String,Person> ptmap;
  ptmap = grid.GetGridMapPessimisticTx<String,Person>("PERSON");
  ptmap.Transaction.Begin();
  Person p = ptmap.GetAndLock("John Doe", LockMode.Upgradable);
  p.age = 30;
  ptmap.Replace(p.name, p);
  ptmap.Transaction.Commit();
  ```

  L'application utilise normalement la méthode GetAndLock plutôt que la simple méthode Get pour verrouiller un enregistrement. La méthode doit être appelée pour fournir la valeur mise à jour à la mappe. Si la méthode Replace n'est pas appelée, la mappe n'est pas modifiée.

Rubrique parent : .NET Programmation de transactions dans des applications .NET

Concepts associés:
Accès aux données et transactions

.NET

Configuration et mise en oeuvre du verrouillage dans les applications .NET

Pour les mappes de sauvegarde auxquelles vous accédez à partir de WebSphere eXtreme Scale Client for .NET, vous devez définir une stratégie de verrouillage pessimiste. Vous pouvez également remplacer le délai de verrouillage d'une instance de mappe. Une fois le verrouillage configuré, vous pouvez verrouiller des clés individuelles ou une liste de clés de la mappe.

Avant de commencer

- Déterminez la stratégie de verrouillage à utiliser. Pour plus d'informations, voir Stratégies de verrouillage.
- Configurez une stratégie de verrouillage pessimiste sur une mappe dynamique. Pour plus d'informations, voir Configuration d'une stratégie de verrouillage.

Procédure

1. Configurez une stratégie de verrouillage pessimiste dans la mappe de sauvegarde. WebSphere eXtreme Scale Client for .NET ne prend en charge que la stratégie de verrouillage pessimiste. Pour plus d'informations, voir Configuration d'une stratégie de verrouillage.
2. Remplacez le délai d'attente de verrou d'une seule instance IGridMapPessimisticTx. Utilisez la propriété **`IGridMapPessimisticTx.LockTimeout`** pour remplacer le délai de verrouillage d'une instance IGridMapPessimisticTx spécifique. La nouvelle valeur du délai de verrouillage affecte toutes les transactions démarrées après sa définition. Cette méthode peut être utilisée lorsque des conflits de verrouillage sont possibles ou prévisibles dans les transactions select.
3. Verrouillez des clés individuelles ou une liste de clés de la mappe. Utilisez la méthode Lock pour verrouiller la clé dans la grille de données ou verrouiller la clé et déterminer si la valeur existe dans la grille de données.
   - La méthode suivante verrouille la clé dans la mappe et renvoie la valeur true si la clé existe ou la valeur false si la clé n'existe pas :

     ```
     bool IGridMapPessimisticTx.Lock(Tkey key, LockMode lockMode);
     ```

   - La méthode suivante verrouille une liste de clés dans la mappe et renvoie une liste de valeurs true ou false ; `true` si la clé existe, `false` si elle n'existe pas :

     ```
     IList<bool> IGridMapPessimisticTx.LockAll(IList<TKey> keyList, LockMode
     lockMode);
     ```

     LockMode est une énumération qui vous permet d'indiquer les clés que vous souhaitez verrouiller à l'aide des valeurs possibles suivantes :

     - Partagé (Shared), pouvant être mis à niveau (Upgradeable) et exclusif (Exclusive)

Voici un exemple de définition du paramètre LockMode :

```
ptmap.Transaction.Begin();
ptmap.Lock(key, LockMode.Upgradable);
ptmap.Put(key,value);
ptmap.Transaction.Commit();
```

Rubrique parent : .NET Programmation de transactions dans des applications .NET

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
.NET Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET
Configuration d'une stratégie de verrouillage

.NET

Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET

Pour éviter que les verrous soient maintenus trop longtemps lorsqu'une exception LockTimeoutException ou une exception LockDeadlockException se produit, votre application doit intercepter les exceptions inattendues et appeler la méthode rollback lorsqu'un événement imprévu se produit.

Procédure

1. Intercepter l'exception et afficher le message résultant.

   ```
   try {
   ...
   } catch (GridException ge) {
      System.Console.WriteLine(ge.ToString());
   }
   ```

   Lorsqu'une exception LockDeadlockException est émise, elle peut être contenue comme exception interne dans une autre exception. Le fragment de code ci-dessus affiche l'exception de niveau supérieur ainsi que, le cas échéant, la totalité de la chaîne de l'exception interne. Le message d'exception propre à l'exception LockDeadlockException contient des détails concernant le conflit de verrouillage. Pour plus d'informations sur la manière d'interpréter ce message, voir Traitement des problèmes d'interblocage.

   ```
   IBM.WebSphere.Caching.Map.LockDeadlockException: Message
   ```

   Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

2. Annuler la transaction après une exception :

   ```
   IGridMapPessimisticTx<String,Person> ptmap;
   ptmap = grid.GetGridMapPessimisticTx<String,Person>(“PERSON”);
   try {
      ptmap.Transaction.Begin();
      Person p = ptmap.Get("Lynn");
      // Lynn a fêté son anniversaire, donc nous l'avons vieilli d'1 an.    p.Age++;
      ptmap.Put(p.name, p);
      ptmap.Transaction.Commit();
   }
   catch (GridException ge) {
      System.Console.WriteLine(ge.ToString());
   }
   finally {
      if ( ptmap.Transaction.Active )
         ptmap.Transaction.Rollback();
   }
   ```

   Le bloc `finally` du fragment de code garantit qu'une transaction est annulée lorsqu'une exception inattendue se produit. Il ne gère pas seulement une exception de type LockDeadlockException mais également les exceptions imprévues éventuelles. Il traite le cas où une exception est émise lors d'un appel de la méthode commit. Cet exemple ne constitue pas le seul moyen pour traiter les exceptions inattendues ; il existe des cas où une application souhaite intercepter certaines des exceptions inattendues susceptibles de se produire afin de pouvoir afficher l'une de ses exceptions d'application. Vous pouvez ajouter les blocs `catch` dont vous avez besoin, mais l'application doit veiller à ce que le fragment de code ne quitte pas sans terminer la transaction.

Rubrique parent : .NET Programmation de transactions dans des applications .NET

Concepts associés:
Types de verrou
Stratégies de verrouillage
Interblocages

Tâches associées:
.NET Configuration et mise en oeuvre du verrouillage dans les applications .NET
Configuration d'une stratégie de verrouillage

.NET

Configuration de la sécurité de grille de données pour WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer .NET et Java™ pour communiquer via SSL (Secure Sockets Layer) et utiliser la logique d'authentification UserPassword.

Avant de commencer

Vous devez disposer des fichiers `key.jks` et `trust.jks` correspondant à votre environnement.

Procédure

Activez et configurez la sécurité dans les serveurs. Si la sécurité n'est pas encore configurée sur les serveurs, vous pouvez procédez comme suit pour la configurer avec l'exemple d'authentificateur externe.

a. Obtenez les exemples de fichiers de sécurité. Téléchargez les exemples de fichiers dans le fichier `security_extauth.zip` à partir du wiki consacré à WebSphere eXtreme Scale.
   - `xsjaas3.config` : définit la configuration JAAS (Java Authentication and Authorization Service).
   - `sampleKS3.jks` : contient le fichier de clés des valeurs utilisateurs et mot de passe JAAS.
   - `security3.xml` : définit l'authentificateur à utiliser pour la sécurité.

b. Modifiez le fichier `xsjaas3.config` et définissez le chemin d'accès au fichier `sampleKS3.jks`.

c. Si vous voulez générer votre propre fichier de clés privées au lieu d'utiliser le modèle de fichier `sampleKS3.jks`, utilisez l'utilitaire **keytool** pour générer la clé privée.

```
keytool -genkey -alias myalias -keysize 2048 -keystore key.jks -keyalg rsa -dname
"CN=www.mydomain.com" -storepass password -keypass password -validity 3650
```

d. Modifiez `sampleServer.properties` pour activer la sécurité. Le fichier `sampleServer.properties` se trouve dans le répertoire *racine_install_wxs*`\properties`. Supprimez la mise en commentaire et modifiez les valeurs de propriété suivantes :

```
securityEnabled=true
secureTokenManagerType=none
alias=ogsample
contextProvider=IBMJSSE2
protocol=SSL
keyStoreType=JKS
keyStore=../../../../xio.test/etc/test/security/key.jks
keyStorePassword=ogpass
trustStoreType=JKS
trustStore=../../../../xio.test/etc/test/security/trust.jks
trustStorePassword=ogpass
```

Que faire ensuite

Configurez TLS (Transport Layer Security) pour WebSphere eXtreme Scale Client for .NET. Pour plus d'informations, voir Configuration de TLS pour WebSphere eXtreme Scale Client for .NET.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

.NET

Configuration de TLS pour WebSphere eXtreme Scale Client for .NET

Vous pouvez configurer Transport Layer Security (TLS) pour WebSphere eXtreme Scale Client for .NET.

Avant de commencer

- Vous devez disposer d'un fichier de clés comportant les mots de passe associés que vous voulez ajouter à la configuration de WebSphere eXtreme Scale Client for .NET.

Procédure

1. Facultatif : A l'aide de l'utilitaire keytool, extrayez le certificat public du fichier `key.jks`.

   ```
   keytool -export -alias myalias -keystore key.jks -file public.cer -storepass
   password
   ```

   Importez cette clé publique vers le magasin de certificats Windows avec l'outil de gestion des certificats, `certmgr.msc`, pour importer la clé dans le dossier des certificats 'Trusted Root Certification Authority' ou 'Trusted People'. (La propriété **keyStore** du fichier `client.properties` pointe vers ce fichier.)

2. Modifiez le fichier `Client.Net.properties` pour y placer les valeurs de propriété suivantes :

   ```
   securityEnabled=true
   credentialAuthentication=supported
   authenticationRetryCount=3
   credentialGeneratorAssembly=IBM.WebSphere.Caching.CredentialGenerator,Version=8.6.0.
   0,
   Culture=neutral,PublicKeyToken=b439a24ee43b0816
   credentialGeneratorProps=manager manager1
   transportType=ssl-required
   publicKeyFile=<name>.cer
   ```

   La valeur de la propriété credentialGeneratorProps, `manager manager1`, est utilisée comme nom d'utilisateur et mot de passe envoyés au serveur dans l'objet Credential.

   La propriété **publicKeyFile** est affectée d'un chemin relatif d'accès à l'environnement d'exécution .NET. Si cette propriété n'est pas définie, le fichier `public.cer` est recherché dans le magasin de certificats Windows. Si elle est définie, le fichier spécifié est utilisé comme fichier de certificat public SSL. Si le fichier spécifié est introuvable, le client .NET tente de trouver un fichier `public.cer` correspondant dans le magasin de certificats.

3. Facultatif : Codez la valeur de la propriété credentialGeneratorProps. Pour ce faire, transférez votre fichier `Client.Net.properties` sur un ordinateur équipé d'une installation Java de WebSphere eXtreme Scale Client . Exécutez l'utilitaire **FilePasswordEncoder** pour coder la propriété credentialGeneratorProps :

   ```
   FilePasswordEncoder.bat Client.Net.Properties credentialGeneratorProps
   ```

   Lorsque vous exécutez cet utilitaire sur un fichier de propriétés, tous les commentaires contenus dans le fichier sont supprimés.

4. Copiez le fichier *net_client_home*`\IBM.WebSphere.Caching.CredentialGenerator.dll` vers le répertoire *net_client_home*`\sample\SimpleClient\bin\<`*nom_configuration*`>`.
5. Générez l'exemple avec le contexte de projet *nom_configuration*. Exécutez l'exemple sur le serveur.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

.NET

Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Pour envoyer des données d'identification de WebSphere eXtreme Scale Client for .NET au serveur, vous devez implémenter les interfaces ICredentialGenerator et ICredential. Ces interfaces génèrent un objet de données d'identification qui est envoyé à la grille de données et interprété sur le serveur. Sur le serveur, cet objet est interprété par le plug-in correspondant.

Pourquoi et quand exécuter cette tâche

Pour exécuter l'authentification, l'application .NET doit implémenter les interfaces suivantes :

- ICredential : Credential représente les données d'identification d'un client, telles qu'une paire ID utilisateur et mot de passe.
- ICredentialGenerator: CredentialGenerator représente une fabrique de données d'identification pour générer les données d'identification.

Lorsqu'une application client .NET se connecte à un serveur qui nécessite l'authentification, le client doit fournir des données d'identification. Les données d'identification d'un client sont représentées par l'interface ICredential. Ces données peuvent être une paire nom-mot de passe, un ticket Kerberos, un certificat client ou des données au format convenu entre le client et le serveur. Cette interface définit explicitement les méthodes equals(Object) et hashCode. Ces deux méthodes sont importantes car les objets Subject authentifiés sont mis en cache par le biais de l'objet Credential en tant que clé sur le serveur. Vous pouvez également générer des données d'identification avec l'interface ICredentialGenerator. Cette interface est utile lorsque les données d'identification peuvent expirer. De nouvelles données d'identification sont générées chaque fois que la propriété Credential est obtenue.

Vous pouvez également utiliser le plug-in d'accréditation CredentialGenerator fourni pour créer des données d'identification basées sur le paramètre **credentialGeneratorProps=** du fichier `Client.Net.Properties`. Les autres paramètres qui définissent le plug-in d'accréditation sont **credentialGeneratorAssembly** et **credentialGeneratorClass**.

Procédure

Implémentez les interfaces ICredentialGenerator et ICredential dans votre application .NET.

A faire : Si vous implémentez ce plug-in d'accréditation du client, vous devez également implémenter un plug-in d'accréditation de serveur capable de recevoir et d'interpréter les données d'identification de WebSphere eXtreme Scale Client for .NET.

Vous pouvez utiliser les exemples suivants pour développer votre application :

- Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET
- Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET

.NET **Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET**
Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredential. Les données d'identification utilisateur/mot de passe stockent un ID utilisateur et un mot de passe.

.NET **Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET**
Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredentialGenerator. L'interface utilise un ID utilisateur et un mot de passe. L'objet UserPasswordCredential contient l'ID utilisateur et le mot de passe obtenus de la propriété Credential accessible en lecture seule.

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Référence associée:
.NET Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET
.NET Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET
Fichier de propriétés du client

Information associée:
Interface ICredential
Interface ICredentialGenerator

.NET

Exemple : Implémentation des données d'identification d'utilisateur et de mot de passe pour les applications .NET

Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredential. Les données d'identification utilisateur/mot de passe stockent un ID utilisateur et un mot de passe.

UserPasswordCredential.cs

```
// Module  :  UserPasswordCrediential.cs

using System;
using IBM.WebSphere.Caching.Security;

namespace com.ibm.websphere.objectgrid.security.plugins.builtins
{
    public class UserPasswordCredential : ICredential
    {
        private String ivUserName;

        private String ivPassword;

        /// <summary>
        ///Creates a UserPasswordCredential with the specified user name and
        /// password.
        ///
        /// ArgumentException if userName or password is null
        /// </summary>
        /// <param name="userName">the user name for this credential</param>
        /// <param name="password">the password for this credential</param>
        public UserPasswordCredential(String userName, String password)
        {
            if (userName == null || password == null) {
                throw new ArgumentException("User name and password cannot be null.");
            }
            this.ivUserName = userName;
            this.ivPassword = password;
        }

        /// <summary>Obtention du nom d'utilisateur de ces données d'identification.
</summary>
        /// <returns>retourne l'argument de nom d'utilisateur envoyé au constructeur
        ///ou la méthode setUserName(String) de cette classe  </returns>
        public String GetUserName() {
            return ivUserName;
        }

        /// <summary>Définir le nom d'utilisateur de ces données d'identification.
        ///ArgumentException if userName is null
        /// </summary>
        /// <param name="userName">userName the user name to set.</param>
        public void SetUserName(String userName) {
            if (userName == null) {
                throw new ArgumentException("User name cannot be null.");
            }
            this.ivUserName = userName;
        }

        /// <summary>Gets the password for this credential.
        /// </summary>
        /// <returns>retourne l'argument de mot de passe envoyé au constructeur ou la
méthode setPassword(String) de cette classe</returns>
        public String GetPassword() {
            return ivPassword;
        }

        /// <summary>Définir le mot de passe de ces données d'identification.
        ///ArgumentException if password is null
```

```
        /// </summary>
        /// <param name="password">the password to set.</param>
        public void SetPassword(String password) {
            if (password == null)
            {
                throw new ArgumentException("Password cannot be null.");
            }
            this.ivPassword = password;
        }

        /// <summary>Vérifie deux objets UserPasswordCredential pour l'égalité.
        ///<p>
        /// Deux objets UserPasswordCredential sont égaux si et seulement si leurs noms
d'utilisateur et mots de passe
        /// sont égaux.
        /// </summary>
        /// <param name="o">Objet dont nous vérifions l'égalité avec cet objet.</param>
        /// <returns> retourne true sur les deux objets UserPasswordCredential sont
équivalents.</returns>
        public bool Equals(ICredential credential)
        {
            if (this == credential) {
                return true;
            }
            if (credential is UserPasswordCredential) {
                UserPasswordCredential other = (UserPasswordCredential)credential;
                return other.ivPassword.Equals(ivPassword) &&
other.ivUserName.Equals(ivUserName);
            }
            return false;
        }

        /// <summary>Retourne le code de hachage de l'objet UserPasswordCredential.
        /// </summary>
        /// <returns>retourne le code de hachage de cet objet</returns>
        public override int GetHashCode() {
            int ret = ivUserName.GetHashCode() + ivPassword.GetHashCode();
            return ret;
        }

        /// <summary>this.Objet comme chaîne
        /// </summary>
        /// <returns>retourne la présentation de chaîne de l'objet UserPasswordCredential
object.</returns>
        public override String ToString() {
            return typeof(UserPasswordCredential).FullName + "[" + ivUserName +
",xxxxxx]";
        }

    }
}
```

Rubrique parent : .NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Information associée:
Interface ICredential
Interface ICredentialGenerator

.NET

Exemple : Implémentation d'un générateur de données d'identification pour les applications .NET

Vous pouvez utiliser cet exemple pour écrire votre propre implémentation de l'interface ICredentialGenerator. L'interface utilise un ID utilisateur et un mot de passe. L'objet UserPasswordCredential contient l'ID utilisateur et le mot de passe obtenus de la propriété Credential accessible en lecture seule.

UserPasswordCredentialGenerator.cs

```
// Module  :  UserPasswordCredientialGenerator.cs
//
// Source File Description: Reference Documentation
//
using System;
using System.Security.Authentication;
using IBM.WebSphere.Caching.Security;
using com.ibm.websphere.objectgrid.security.plugins.builtins;

namespace IBM.WebSphere.Caching.Security
{
    public class UserPasswordCredentialGenerator : ICredentialGenerator
    {

        private String ivUser;

        private String ivPwd;

        public ICredential Credential { get { return _getCredential(); } }

        public string Properties { set {_setProperties(value);} }

        public UserPasswordCredentialGenerator()
        {
            ivUser = null;
            ivPwd = null;
        }


        public UserPasswordCredentialGenerator(String user=null, String pwd=null)
        {
            ivUser = user;
            ivPwd = pwd;
        }

        /// <summary>Crée un objet UserPasswordCredential en utilisant le nom
d'utilisateur et le mot de passe de l'objet.
        /// </summary>
        /// <returns>new UserPasswordCredential instance</returns>
        private ICredential _getCredential()
        {
            try
            {
                ICredential MyCredential = new UserPasswordCredential(ivUser, ivPwd) as
ICredential;
                return (ICredential) MyCredential;
            }
            catch (Exception e)
            {
                AuthenticationException CannotGenerateCredentialException = new
AuthenticationException(e.ToString());
                throw CannotGenerateCredentialException;
            }
        }

        /// <summary>Obtient le mot de passe du générateur de données d'identification.
        /// </summary>
        /// <returns> retourne l'argument de mot de passe envoyé au constructeur</returns>
```

```
        public String getPassword()
        {
            return ivPwd;
        }

        /// <summary>Obtient le nom d'utilisateur de ces données d'identification.
        /// </summary>
        /// <returns>retourne l'argument de nom d'utilisateur envoyé au constructeur de
cette classe</returns>
        public String getUserName()
        {
            return ivUser;
        }

        /// <summary>Définit des propriétés supplémentaires, à savoir un nom d'utilisateur
et un mot de passe.
        ///émet une exception ArgumentException si le format n'est pas valide
        /// </summary>
        /// <param name="properties">properties chaîne de propriétés avec un nom
d'utilisateur et un mot de passe séparés par un espace.</param>
        private void _setProperties(string properties)
        {
            String token = properties;
            char[] Seperator = { ' ' };
            String[] StringProperty = properties.Split(Seperator);
            if (StringProperty.Length != 2)
            {
                throw new ArgumentException(
                    "The properties should have a user name and password and separated by
a space.");
            }

            ivUser = StringProperty[0];
            ivPwd = StringProperty[1];
        }

        /// <summary>Vérifie l'égalité de deux objets UserPasswordCredentialGenerator.
        ///<p>
        ///Deux objets UserPasswordCredentialGenerator sont égaux si et seulement si leurs
noms d'utilisateur et mots de passe
        ///sont égaux..
        /// </summary>
        /// <param name="obj">Objet dont nous vérifions l'égalité avec cet objet.</param>
        /// <returns><code>true</code> si les deux objets  UserPasswordCredentialGenerator
sont équivalents</returns>
        public override bool Equals(Object obj)
        {
            if (obj == this)
            {
                return true;
            }

            if (obj != null && obj is UserPasswordCredentialGenerator)
            {
                UserPasswordCredentialGenerator other = (UserPasswordCredentialGenerator)
obj;

                Boolean bothUserNull = false;
                Boolean bothPwdNull = false;

                if (ivUser == null)
                {
                    if (other.ivUser == null)
                    {
                        bothUserNull = true;
                    }
                    else
                    {
```

```
                        return false;
                    }
                }

                if (ivPwd == null)
                {
                    if (other.ivPwd == null)
                    {
                        bothPwdNull = true;
                    }
                    else
                    {
                        return false;
                    }
                }
                return (bothUserNull || ivUser.Equals(other.ivUser)) && (bothPwdNull ||
ivPwd.Equals(other.ivPwd));
            }
            return false;
        }

        /// <summary>Retourne le code de hachage de l'objet
UserPasswordCredentialGenerator.
        /// </summary>
        /// <returns>Retourne le code de hachage de cet objet</returns>
        public override int GetHashCode()
        {
           return ivUser.GetHashCode() + ivPwd.GetHashCode();
        }

    }

}
```

Rubrique parent : .NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Tâches associées:
.NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Information associée:
Interface ICredential
Interface ICredentialGenerator

.NET

Programmation de données d'identification personnalisées pour WebSphere eXtreme Scale Client for .NET

Vous pouvez spécifier des données d'identification de l'utilisateur pour une mappe. De cette manière, deux utilisateurs peuvent interagir avec la même grille de données par l'intermédiaire d'une application Web.

Procédure

1. Définissez les données d'identification de l'utilisateur dans le fichier `client.properties`.

```
credentialAuthentication=required
authenticationRetryCount=3
credentialGeneratorAssembly=IBM.WebSphere.Caching.CredentialGenerator,
Version=8.6.0.0, Culture=neutral, PublicKeyToken=b439a24ee43b0816
credentialGeneratorClass=IBM.WebSphere.Caching.Security.UserPasswordCredentialGenera
tor
credentialGeneratorProps=manager manager1
```

2. Ajoutez une référence au fichier `IBM.WebSphere.Caching.CredentialGenerator.dll` dans votre projet d'application. Cette DLL de plug-in contient l'implémentation ICredentialGenerator.
3. Utilisez les API ICredentialGenerator dans votre application .NET.

```
//GridManagerFactory.GetGridManager
IGridManager gm = GridManagerFactory.GetGridManager();
//IGridManager.Connect
ICatalogDomainInfo cdi =
gm.CatalogDomainManager.CreateCatalogDomainInfo(hostAndPort);
ctx = gm.Connect(cdi, "client.properties");
//IGridManager.GetGrid
IGrid grid = gm.GetGrid( ctx, "Grid");
ICredentialGenerator credGenManager = new UserPasswordCredentialGenerator("manager",
"manager1");
ICredentialGenerator credGenOperator = new
UserPasswordCredentialGenerator("operator", "operator1");
//IGrid.GetGridMap
IGridMapPessimisticAutoTx<Object, Object> gridMap1  =
grid.GetGridMapPessimisticAutoTx<Object, Object>("Map1", credGenManager);

IGridMapPessimisticAutoTx<Object, Object> gridMap2  =
grid.GetGridMapPessimisticAutoTx<Object, Object>("Map1", credGenOperator);
```

Rubrique parent : .NET 2.5+ Développement d'applications de grille de données avec les API .NET

Surveillance

Vous pouvez contrôler plusieurs aspects de WebSphere DataPower XC10 Appliance, notamment l'état des changements de configuration via la vue **Tâches**, ainsi que les performances de vos grilles de données via la vue **Contrôler** de l'interface utilisateur.

Avant de commencer

Pour afficher des données dans la vue **Tâches** ou la vue **Contrôler** de l'interface utilisateur, votre dispositif doit être configuré et vous devez avoir créé des grilles de données recevant de nouvelles entrées en provenance des applications.

2.5+ **Suivi de l'état de démarrage du dispositif**
Vous pouvez surveiller l'état de démarrage du dispositif. Cela peut vous aider à identifier et résoudre les problèmes éventuels.

Surveillance des grilles de données dans l'interface utilisateur
Vous pouvez utiliser les fonctionnalités de création de graphiques de WebSphere DataPower XC10 Appliance pour afficher les performances globales des grilles de données de votre environnement.

Contrôle des activités à l'aide des tâches
Les tâches permettent de contrôler la progression des modifications administratives (par exemple, les ajouts de dispositif à la collectivité).

Surveillance avec l'utilitaire xscmd
Avec l'utilitaire **`xscmd`**, vous pouvez afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre dispositif.

Surveillance à l'aide de fichiers CSV
Les données de surveillance sont automatiquement consignées dans des fichiers CSV. Ces fichiers CSV peuvent contenir des informations sur les serveurs, la mappe ou la grille de données.

Surveillance avec l'utilitaire xsadmin
Avec l'utilitaire **`xsadmin`**, vous pouvez mettre en forme et afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre WebSphere DataPower XC10 Appliance. Il fournit une méthode pour effectuer l'analyse syntaxique et la détection des données de déploiement actuelles et peut servir de base pour l'écriture d'utilitaires personnalisés.

2.5+ **Surveillance de la santé de l'environnement**
Message Center fournit une vue agrégée des notifications d'événements pour les messages de journal et de l'outil de diagnostic de premier niveau. Vous pouvez afficher ces notifications d'événements avec : Message Center dans la console Web, l'utilitaire **`xscmd`**, les fichiers journaux de santé, l'interface de commande HTTP, ou un programme avec MBeans.

Surveillance avec SNMP (Simple Network Monitoring Protocol)
Avec le support SNMP (Simple Network Monitoring Protocol), vous pouvez contrôler l'état d'un dispositif IBM® WebSphere DataPower XC10 Appliance dans le cadre d'un grand groupe de systèmes dans un centre de données. La surveillance SNMP améliore votre capacité de notifier les problèmes et de les résoudre rapidement.

Configuration de la consignation à distance
Vous pouvez activer la consignation à distance pour enregistrer les entrées de journal sur un serveur distant en dehors du dispositif. La consignation à distance peut être utile lorsque vous devez définir un niveau de journal de débogage détaillé pour isoler un problème ou surveiller un comportement sur une longue période.

Suivi de l'état de démarrage du dispositif

2.5+ Vous pouvez surveiller l'état de démarrage du dispositif. Cela peut vous aider à identifier et résoudre les problèmes éventuels.

Avant de commencer

- Initialisez, démarrez ou redémarrez le dispositif.
- Vous devez avoir accès au nom d'utilisateur et au mot de passe de `xcadmin`.

Pourquoi et quand exécuter cette tâche

Pour suivre l'état de démarrage du dispositif, vous pouvez utiliser le panneau de démarrage de l'interface utilisateur ou la commande **`start-progress`**.

Panneau de l'interface utilisateur consacré au démarrage du dispositif

Ce panneau affiche l'état de démarrage, sous la forme d'un pourcentage d'achèvement et de l'état des différents serveurs et processus. Il est actualisé automatiquement toutes les 15 secondes.

Commande `start-progress`

La commande **`start-progress`** s'exécute à l'aide de l'interface de ligne de commande. Elle renvoie l'état en cours du démarrage. Pour obtenir une version actualisée de cet état, vous devez exécuter à nouveau la commande.

Procédure

- Affichage du panneau d'état de démarrage du dispositif dans l'interface utilisateur.
  1. Utilisez l'URL suivante pour accéder au panneau d'état de démarrage du dispositif :

     ```
     https://<nom_hôte_dispositif>:9443/
     ```

     Le pourcentage d'avancement du démarrage s'affiche.
  2. Si la progression reste bloquée à un certain pourcentage, vous pouvez télécharger les fichiers journaux pour diagnostiquer l'origine du problème. Cliquez sur **Télécharger les fichiers journaux**.
- Affichage de l'état de démarrage du dispositif dans l'interface de ligne de commande.
  1. Connectez-vous à l'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.
  2. Exécutez la commande **`start-progress`**. L'état de la séquence de démarrage du dispositif s'affiche.
  3. Pour actualiser cet état, exécutez de nouveau la commande **`start-progress`**.

Exemple

L'exemple suivant illustre une séquence de démarrage réussie :

```
Le montage du volume 1 a abouti.
Le montage du volume 2 a abouti.
Le service d'administration de grille a démarré.
Le serveur de catalogue a démarré.
Le service de configuration de grille a démarré.
Le serveur de conteneur 01 a démarré.
Le serveur de conteneur 02 a démarré.
Le serveur de conteneur 03 a démarré.
Le serveur de conteneur 04 a démarré.
Le serveur de conteneur 05 a démarré.
Le serveur de conteneur 06 a démarré.
Le serveur de conteneur 07 a démarré.
Le serveur de conteneur 08 a démarré.
La console d'administration a démarré.
```

- Les serveurs de catalogue s'exécutent sur les trois premiers dispositifs de la collectivité. Il se peut que vous ne voyiez pas le message de démarrage du serveur de catalogue si vous utilisez plus de trois dispositifs.
- Le nombre de serveurs de conteneur qui démarrent dépend du type de transport que vous utilisez. Avec IBM eXtremeIO (XIO), 8 serveurs de conteneur démarrent par défaut. Avec ORB (Object Request

Broker), 16 serveurs de conteneur démarrent par défaut. Pour plus d'informations sur les types de transport, voir Configuration d'IBM eXtremeIO (XIO).
- Après l'affichage du message d'état `La console d'administration a démarré.`, vous pouvez vous connecter à l'interface utilisateur.

Que faire ensuite

- Le panneau de démarrage du dispositif indique également l'état Suspendu lorsque vous exécutez les commandes **`request suspend`** et **`request force-suspend`**.

Rubrique parent : Surveillance

Surveillance des grilles de données dans l'interface utilisateur

Vous pouvez utiliser les fonctionnalités de création de graphiques de WebSphere DataPower XC10 Appliance pour afficher les performances globales des grilles de données de votre environnement.

Avant de commencer

Une fois les grilles de données créées et les applications configurées pour les utiliser, les statistiques deviennent disponibles au bout d'un certain temps. Par exemple, avec une grille de données de mémoire cache dynamique, les statistiques deviennent disponibles lorsqu'un WebSphere Application Server exécutant un système de mémoire cache dynamique se connecte à la grille de données de mémoire cache dynamique du dispositif. Si vous utilisez une collectivité, l'initialisation de cette collectivité doit être terminée pour que les statistiques soient disponibles. En général, il suffit d'attendre environ une minute après une modification de configuration importante pour observer le changement au niveau des statistiques.

Pourquoi et quand exécuter cette tâche

Comportement des fonctions graphiques

La série de données Jour, Semaine et Mois persiste à long terme. En revanche, la série de données **Dernière heure** n'est stockée qu'en mémoire. Les données de dernière heure sont stockées aux emplacements suivants :

- En mémoire pour une grille de données
- En mémoire pour le cache de console

Si des données statistiques sont perdues dans un seul de ces emplacements, les données sont toujours disponibles. Si elles sont perdues aux deux emplacements, les données n'apparaissent pas dans le graphique.

Cas de perte des données en mémoire :

- Lors du déplacement de grilles de données, il y a des périodes pendant lesquelles aucune statistique n'est accumulée.
- Lorsque les dispositifs sont redémarrés, les données en mémoire de la grille de données et dans le cache de console sont effacées.
- Lorsque la mémoire du cache de console se remplit, les données utilisées le moins récemment sont effacées de ce cache. Le cache de console contient 2048 entrées de statistiques.
- Lorsque la console est inactive pendant 30 minutes, le processus de console redémarre et la mémoire est effacée.

La collecte des statistiques se poursuit pour la série de données de la dernière heure.

Conseil : Il suffit de déplacer le pointeur de la souris sur n'importe quel point de données du graphique pour afficher des informations plus précises sur ce point.

Procédure

- Pour afficher les performances de toutes vos grilles de données, cliquez sur **Contrôler** > **Généralités sur la grille de données**. Cette page contient les informations suivantes :

  Onglet Capacité utilisée

  Distribution de la capacité actuelle utilisée par les grilles de données

  Ce graphique contient une vue de la capacité totale disponible sur le dispositif ou la collectivité, de la capacité totale utilisée par la copie principale et par les répliques des données, de la capacité limitée restante si une limite de capacité a été définie et des grilles de données qui consomment la partie la plus importante de la capacité. Vous pouvez utiliser les options ci-dessous pour trier les données. Ne sont affichées que les premières 25 grilles de données :

  - Consommatrices de la plus grande capacité
  - Pourcentage le plus élevé de capacité limitée utilisée

  Capacité utilisée sur une période donnée

  Ce graphique représente la capacité utilisée au cours de la période sélectionnée.

Onglet Débit moyen

Les 5 grilles de données les plus actives du point de vue du débit moyen en transactions/seconde

Ce graphique contient la liste des 5 principales grilles de données, classées par débit moyen, mesuré en transactions par seconde.

Débit moyen sur une période donnée

Ce graphique représente le débit moyen, mesuré en transactions par seconde, au cours de la période sélectionnée.

Onglet Délai de transaction moyen sur une période donnée

Les 5 grilles de données les plus lentes du point de vue du délai de transaction moyen sur une période donnée, en millisecondes

Ce graphique contient la liste des cinq grilles de données les plus lentes, classés par délai de transaction moyen sur une période donnée.

Délai de transaction moyen sur une période donnée

Ce graphique représente la durée moyenne des transactions au cours de la période sélectionnée.

- Pour afficher les différentes grilles de données, cliquez sur **Contrôler** > **Généralités sur des grilles de données spécifiques** > ***nom_grille_données***. Cette page affiche un récapitulatif incluant le nombre d'entrées en cache, le délai de transaction moyen, le débit moyen, le taux de réussite en mémoire cache et le pourcentage de capacité limitée au cours des 30 dernières secondes. Vous pouvez également afficher les graphiques suivants :

Capacité utilisée

Ce graphique montre la capacité utilisée du cache par rapport au nombre d'entrées et la limite de capacité configurée du cache. La capacité utilisée porte sur les données primaires et les données répliquées. Vous pouvez modifier l'intervalle affiché : dernière heure, dernier jour, dernière semaine, dernier mois. Le niveau de détail affiché sur le graphique varie en fonction de l'intervalle sélectionné.

Interprétation du nombre d'entrées de la mémoire cache :

- **Pour toutes les grilles de données :** le nombre d'entrées de cache ne représente que les fragments primaires. Pour afficher le nombre d'entrées de cache pour les fragments primaires et les fragments répliques, utilisez la commande **`xscmd -c showMapSizes`**.
- **Pour les grilles de données de mémoire cache dynamique :** 166 entrées de mémoire cache sont créées par défaut pour chaque grille de données de mémoire cache dynamique. Chaque grille de mémoire cache dynamique comporte 83 partitions, et chaque partition ou fragment est initialisé avec deux entrées pour les grilles de mémoire cache dynamique. Par conséquent, le nombre d'entrées de mémoire cache lors de l'initialisation est de 166. Ces entrées de mémoire cache contiennent les statistiques du fournisseur de mémoire cache dynamique et la configuration de la mémoire cache dynamique pour WebSphere Application Server. Par conséquent, 166 entrées de mémoire cache sont affichées dans le panneau de surveillance avant que vous n'ajoutiez des données à la grille de données de mémoire cache dynamique.
- **Pour les grilles de données de session :** Le nombre d'entrées de la grille de données inclut le nombre de sessions, le nombre d'attributs entre toutes les sessions et les entrées de la table d'expulsion. La table d'expulsion est alimentée lorsqu'une session arrivée à expiration n'a pas été invalidée par le conteneur Web. La table d'expulsion n'est alimentée que lors d'une reprise en ligne d'un serveur d'applications.

Utilisation du cache

Ce graphique permet de visualiser le nombre de demandes ayant abouti au cache. Vous pouvez afficher les tentatives de cache, les réussites en mémoire cache et le taux de réussite en mémoire cache dans le graphique.

Débit moyen

Ce graphique montre le nombre moyen de transactions par seconde traitée sur une période et la durée moyenne de chaque transaction.

- Pour afficher plus de détails à propos d'une grille de données spécifique, cliquez sur **Contrôler** > **Rapports détaillés sur la grille de données**. Une arborescence affiche toutes les grilles de données de votre configuration. Vous explorer cette arborescence en aval et accéder à une grille de données spécifique afin d'afficher les mappes appartenant à cette grille de données. Vous pouvez cliquer sur un nom de grille de données ou sur une mappe pour obtenir plus d'informations :

Informations une grille de données

Vous pouvez afficher les capacités utilisées ainsi que la liste des zones auxquelles la grille de données appartient. La capacité utilisée porte sur les données primaires et les données répliquées. Le graphique qui indique la capacité consommée par les 25 principales mappes dans la grille de données s'affiche. Vous pouvez également afficher un pool total qui inclut la capacité par zone. Un graphique indique la capacité de grille de données consommée dans les 25 principales zones.

Informations sur une mappe

Vous pouvez afficher plus d'informations concernant les mappes de chaque grille, notamment les

informations suivantes : nombre total d'entrées de cache primaires de la mappe, débit moyen, durée moyenne de la transaction et capacité totale de la mappe ventilée par rapport aux 25 principales partitions.

Rubrique parent : Surveillance

Information associée:
Leçon 4 du tutoriel du guide de démarrage : Surveillance de l'environnement

Contrôle des activités à l'aide des tâches

Les tâches permettent de contrôler la progression des modifications administratives (par exemple, les ajouts de dispositif à la collectivité).

Pourquoi et quand exécuter cette tâche

Vous pouvez suivre l'état d'une tâche du début à la fin de celle-ci. L'état d'une tâche peut être En file d'attente, En cours d'exécution, Succès ou Echec.

Tableau 1. Icônes d'état des tâches

Icône	Description
En file d'attente ( )	La tâche est entrée dans la file d'attente mais son exécution n'a pas commencé.
En cours d'exécution ( )	La tâche est en cours d'exécution.
Succès ( )	La tâche s'est exécutée sans erreur.
Echec ( )	La tâche ne s'est pas exécutée correctement. Pour plus d'informations concernant l'erreur qui s'est produite, voir les messages dans la tâche.

Chaque tâche est constituée d'un ou plusieurs messages qui fournissent plus d'informations sur son état.

Tableau 2. Icônes de message de tâche

Icône	Description
Information ( )	Le message contient des informations sur une étape de la tâche en cours.
Erreur ( )	La tâche ne s'est pas exécutée correctement. Pour plus d'informations concernant l'erreur qui s'est produite, voir les messages dans la tâche.
2.5+ Avertissement ( )	Le message contient des informations concernant un risque d'échec de la tâche. Mais il se peut que la tâche aboutisse.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Tâches**.
2. Pour afficher une tâche spécifique et obtenir plus d'informations à son sujet, cliquez sur le nom de la tâche. Chaque tâche contient des informations telles que son type, ses dates de début et de fin, et la liste des messages d'état spécifiques permettant de surveiller sa progression. L'état est mis à jour toutes les 10 secondes jusqu'à ce que la tâche se termine, correctement ou non.
3. 2.5+ Facultatif : Suppression des tâches terminées. Une tâche terminée est à l'état Succès ou Echec. Cliquez sur **Supprimer toutes les tâches terminées** ( ). Vous êtes alors invité à confirmer que vous voulez supprimer toutes les tâches terminées. Si vous cliquez sur **OK**, toutes les tâches terminées sont supprimées. Les tâches dont l'état est En file d'attente ou En cours d'exécution ne sont pas supprimées.

Rubrique parent : Surveillance

Tâches associées:
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif

Surveillance avec l'utilitaire `xscmd`

Avec l'utilitaire **`xscmd`**, vous pouvez afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre dispositif.

Avant de commencer

- Voir Administration avec l'utilitaire xscmd pour plus d'informations sur le démarrage de l'utilitaire **`xscmd`**.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'utilitaire **`xscmd`** pour afficher la structure et l'état actuels de la grille de données (par exemple, le contenu de la grille. Dans cet exemple, la structure de grille de données dans cette tâche est une grille de données simple *`ObjectGridA`* avec la mappe *`MapA`* qui appartient au groupe de mappes *`MapSetA`*. Cet exemple montre comment afficher tous les conteneurs dans une grille de données et afficher des mesures filtrées concernant la taille de la mappe *`MapA`*. Pour afficher toutes les options de la commande, exécutez l'utilitaire **`xscmd`** sans arguments ou avec l'option **`-help`**.

Procédure

1. Surveillez l'environnement avec l'utilitaire **`xscmd`**.
   - Pour activer les statistiques pour tous les serveurs, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c setStatsSpec -spec ALL=enabled -g ObjectGridA`
     - Windows `xscmd.bat -c setStatsSpec -spec ALL=enabled -g ObjectGridA`
   - Pour afficher tous les serveurs de conteneur en ligne pour une grille de données, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c showPlacement -g ObjectGridA -ms MapSetA`
     - Windows `xscmd.bat -c showPlacement -g ObjectGridA -ms MapSetA`

     Toutes les informations sur les conteneurs s'affichent.

     Avertissement : Pour obtenir ces informations lorsque le protocole TLS/SSL (Transport Layer Security/Secure Sockets Layer) est activé, vous devez démarrer les serveurs de catalogue et de conteneur avec le port de service JMX défini. Pour définir le port du service JMX, vous pouvez utiliser l'option **`-JMXServicePort`** dans le script **`startOgServer`** ou **`startXsServer`** ou bien appeler la méthode setJMXServicePort dans l'interface ServerProperties.

   - Pour afficher des informations sur les mappes de la grille de données ObjectGridA, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA`
     - Windows `xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA`
   - Pour vous connecter au service de catalogue et afficher des informations sur la mappe MapA pour l'ensemble du domaine de service de catalogue, exécutez la commande suivante :
     - UNIX `./xscmd.sh -c showMapSizes -g ObjectGridA -ms MapSetA -m MapA -cep CatalogMachine:6645`
     - Windows `xscmd.bat -c showMapSizes -g ObjectGridA -ms MapSetA -m MapA -cep CatalogMachine:6645`
   - Pour afficher le placement configuré et d'exécution de votre configuration, exécutez la commande suivante :
     - `xscmd -c placementServiceStatus`
     - `xscmd -c placementServiceStatus -g ObjectGridA -ms MapSetA`
     - `xscmd -c placementServiceStatus -ms MapSetA`
     - `xscmd -c placementServiceStatus -g ObjectGridA`

     Vous pouvez définir la portée de la commande pour afficher les informations de placement de l'intégralité de la configuration, une grille de données unique, un groupe de mappes unique ou une combinaison de grille de données et de groupe de mappes
2. Affichez les résumés des états de réplication dans l'environnement.
   - Afficher le résumé des révisions en attente de chaque serveur de conteneur. Vous pouvez exécuter la commande sur un serveur de conteneur spécifique avec l'argument **`-ct`** ou sur tous les serveurs de conteneur si vous n'incluez pas d'argument.
     - UNIX `./xscmd.sh -c showReplicationState -ct container1`
     - Windows `xscmd.bat -c showReplicationState -ct container1`

     Les informations contenues dans la sortie de cette commande inclut la réplication sortante et la réplication entrante. La réplication sortante contient les modifications qui doivent être extraites du fragment primaire sur le serveur de conteneur et placées dans ses fragments réplique sur les autres serveurs de conteneur. La réplication entrante contient les modifications qui doivent être extraites des fragments principaux sur les autres serveur de conteneur et placées dans les

répliques sur le serveur de conteneur. Ces données statistiques peuvent donner une idée de la santé de réplication. Si le nombre de révisions en suspens sur un serveur de conteneur augmente considérablement, des problèmes au niveau du conteneur peuvent exister.

- Affichez le résumé des révisions en attente des fragments entre les domaines de service de catalogue. Vous pouvez exécuter la commande sur un serveur de conteneur spécifique et le domaine de service de catalogue, ou l'intégralité de votre configuration si vous n'incluez pas d'argument.
  - UNIX `./xscmd.sh -c showDomainReplicationState -dom domainA -ct container1`
  - Windows `xscmd.bat -c showDomainReplicationState -dom domainA -ct container1`

  Les informations contenues dans la sortie de cette commande comprend un récapitulatif des révisions en attente de chaque serveur de conteneur pour chaque domaine de service de catalogue lié. La commande renvoie les modifications à répliquer entre chaque fragment primaire et les fragments primaires distants correspondants qui se trouvent dans un autre domaine de service de catalogue.

Rubrique parent : Surveillance

Tâches associées:
Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.

Surveillance à l'aide de fichiers CSV

Les données de surveillance sont automatiquement consignées dans des fichiers CSV. Ces fichiers CSV peuvent contenir des informations sur les serveurs, la mappe ou la grille de données.

Pourquoi et quand exécuter cette tâche

Les données de surveillance peuvent être consignées par défaut dans des fichiers CSV. Vous pouvez télécharger et analyser des données historiques pour les serveurs exécutés sur le dispositif. La collecte des données commence au démarrage des serveurs. Vous pouvez ensuite télécharger les fichiers CSV à tout moment et utiliser les fichiers comme vous le désirez.

Procédure

1. Téléchargez le fichier CSV. Lors du téléchargement des fichiers journaux à partir du dispositif, les fichiers CSV sont inclus dans le fichier `trace.zip`. Pour télécharger ce fichier, cliquez sur **Dispositif** > **Identification et résolution des incidents** > **Consignation au journal** > **Télécharger les fichiers journaux**. Dans le fichier `trace.zip`, les fichiers CSV se trouvent dans le répertoire *`nom_serveur`*`/logs`. Les fichiers sont intitulés : `jvmstats.log`, `mapstats.log` et `ogstats.log`.
2. Importez le fichier CSV dans le programme que vous utilisez pour traiter les données, par exemple, une feuille de calcul.

Définition des statistiques des fichiers CSV
Les fichiers CSV que vous pouvez télécharger pour un serveur comprennent des statistiques que vous pouvez utiliser pour créer des diagrammes d'historique ou d'autres informations.

Rubrique parent : Surveillance

Tâches associées:
Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance

Référence associée:
Définition des statistiques des fichiers CSV

Définition des statistiques des fichiers CSV

Les fichiers CSV que vous pouvez télécharger pour un serveur comprennent des statistiques que vous pouvez utiliser pour créer des diagrammes d'historique ou d'autres informations.

Journal des statistiques JVM (Java virtual machine)

TimeStamp (colonne 1)

Indique la date et l'heure de l'image instantanée des statistiques, prise pour la machine virtuelle Java (JVM).

ServerName (colonne 2)

Indique le nom du serveur de la machine JVM.

Hostname (colonne 3)

Indique le nom de la machine JVM.

FreeMemory (colonne 4)

Indique le nombre d'octets disponibles pour la machine JVM.

MaxMemory (colonne 5)

Indique le nombre maximal d'octets qui peut être attribué pour la machine JVM.

TotalMemory (colonne 6)

Affiche l'utilisation de la mémoire réelle dans l'environnement d'exécution du serveur.

AvailProcs (colonne 7)

Affiche le nombre de processeurs qui sont disponibles pour ce service de catalogue et ses mappes. Pour une stabilité optimale, vous devez exécuter les serveurs à 60 % du chargement de processeur et les segments de mémoire des machines virtuelles Java à 60 % du chargement des segments de mémoire. Les pics peuvent alors pousser l'utilisation du processeur à 80-90 %, mais ce ne doit pas être le niveau habituel d'exécution de vos serveurs.

Journal des statistiques de mappe

TimeStamp (colonne 1)

Indique la date et l'heure de l'image instantanée des statistiques, prise pour la mappe.

MapName (colonne 2)

Indique le nom de la mappe.

OgName (colonne 3)

Indique le nom de la grille de données à laquelle appartient la mappe.

PartitionId (colonne 4)

Indique l'ID de la partition.

MapSetName (colonne 5)

Indique le groupe de mappes auquel appartient la mappe.

HitRate (colonne 6)

Affiche le taux de réussites pour la mappe sélectionnée. Un taux élevé est souhaitable. Le taux de réussite indique la manière dont la grille de données contribue à éviter d'accéder au stockage de persistance.

Count (colonne 7)

Indique le nombre d'entrées dans la grille de données depuis le démarrage du serveur. Par exemple, la valeur 100 indique que l'entrée est le 100e échantillon collecté depuis le démarrage du serveur.

TotalGetCount (colonne 8)

Affiche le nombre total de fois où la mappe a dû accéder au stockage de persistance pour obtenir des données.

TotalHitCount (colonne 9)

Affiche le nombre total de fois où les données demandées ont été trouvées dans la mappe, dispensant de devoir accéder au stockage de persistance.

StartTime (colonne 10)

Indique l'heure à laquelle l'appel a commencé à partir de la dernière réinitialisation des compteurs. Les réinitialisations se produisent lorsque le serveur démarre ou redémarre.

LastCount (colonne 11)

Indique la durée écoulée depuis le dernier échantillon de données.

LastTotalGetCount (colonne 12)

Indique le nombre total actuel d'opérations d'extraction à partir de la mémoire cache moins le nombre d'opérations d'extraction dans la période précédente.

LastTotalHitCount (colonne 13)

Indique le nombre total actuel d'opérations d'extraction à partir de la mémoire cache moins le nombre d'opérations d'extraction dans la période précédente.

UsedBytes (colonne 14)

Affiche la consommation de la mémoire par cette mappe. Les statistiques d'octets utilisés sont exactes uniquement lorsque vous utilisez des objets simples ou le mode de copie COPY_TO_BYTES.

MinUsedBytes (colonne 15)

Affiche le point bas de la consommation de mémoire par ce service de catalogue et ses mappes. Les statistiques d'octets utilisés sont exactes uniquement lorsque vous utilisez des objets simples ou le mode de copie COPY_TO_BYTES.

MaxUsedBytes (colonne 16)

Affiche le point haut de la consommation de mémoire par ce service de catalogue et ses mappes. Les statistiques d'octets utilisés sont exactes uniquement lorsque vous utilisez des objets simples ou le mode de copie COPY_TO_BYTES.

LastUsedBytes (colonne 17)

Indique la valeur UsedBytes en cours moins la valeur UsedBytes à partir de la période de collecte des statistiques précédentes.

SampleLen (colonne 18)

Indique la durée, en millisecondes, de la période d'échantillonnage des données.

Journal de statistiques ObjectGrid

TimeStamp (colonne 1)

Indique la date et l'heure de l'image instantanée des statistiques, prise pour la grille de données.

OgName (colonne 2)

Indique le nom de la grille de données.

PartitionId (colonne 3)

Indique l'ID de la partition.

Count (colonne 4)

Indique le nombre d'entrées dans la grille de données qui ont été collectées depuis le démarrage du serveur. Par exemple, la valeur 100 indique que l'entrée est le 100e échantillon collecté depuis le démarrage du serveur.

Hostname (colonne 5)

Indique le nom d'hôte.

DomainName (colonne 6)

Indique le domaine du service de catalogue auquel la grille de données appartient.

MaxTime (colonne 7)

Affiche pour ce serveur le temps *maximum* qu'a mis une transaction pour s'exécuter.

MinTime (colonne 8)

Affiche pour ce serveur le temps *minimum* qu'a mis une transaction pour s'exécuter.

MeanTime (colonne 9)

Indique le temps moyen passé sur une transaction.

TotalTime (colonne 10)

Affiche pour ce serveur le temps total passé à des transactions depuis l'initialisation du serveur.

AvgTransTime (colonne 11)

Affiche pour ce serveur la durée moyenne que met une transaction pour s'exécuter.

AvgThroughPut (colonne 12)

Affiche le nombre moyen de transactions par seconde pour ce serveur.

SumOfSquares (colonne 13)

Spécifie la somme des carrés pour le temps de transaction. Cette valeur mesure l'écart par rapport à la moyenne à un moment donné.

SampleLen (colonne 14)

Indique la durée, en millisecondes, de la période d'échantillonnage des données.

LastDataSample (colonne 15)

Indique la durée écoulée depuis le dernier échantillon de données.

LastTotalTime (colonne 16)

Indique le temps total actuel moins le temps total précédent de l'échantillonnage de données.

StartTime (colonne 17)

Indique l'heure à laquelle les statistiques ont commencé à être collectées depuis la dernière réinitialisation des données. Les données sont réinitialisées lorsque le serveur redémarre.

Rubrique parent : Surveillance à l'aide de fichiers CSV

Tâches associées:

Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance
Surveillance à l'aide de fichiers CSV

Surveillance avec l'utilitaire `xsadmin`

Avec l'utilitaire **`xsadmin`**, vous pouvez mettre en forme et afficher des informations sous forme de texte à propos des grilles de données qui s'exécutent sur votre WebSphere DataPower XC10 Appliance. Il fournit une méthode pour effectuer l'analyse syntaxique et la détection des données de déploiement actuelles et peut servir de base pour l'écriture d'utilitaires personnalisés.

Avant de commencer

- 2.5+ L'utilitaire **`xsadmin`** est obsolète. Utilisez l'utilitaire **`xscmd`** à la place. L'utilitaire **`xscmd`** est fourni comme utilitaire pris en charge pour la surveillance et l'administration de votre environnement. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Pourquoi et quand exécuter cette tâche

L'utilitaire **`xsadmin`** utilise une implémentation de beans gérés (MBeans). Vous pouvez étendre les capacités de cet utilitaire à l'aide des interfaces du package com.ibm.websphere.objectgrid.management. Vous pouvez consulter le code source de l'application **`xsadmin`** dans le fichier *`rép_base_client_wxs`*`/samples/xsadmin.jar` dans le cas d'une installation autonome, ou bien dans le fichier *`rép_base_client_wxs`*`/optionalLibraries/ObjectGrid/xsadmin.jar` dans le cas d'une installation WebSphere Application Server.

Procédure

1. Téléchargez le fichier de clés certifiées actif pour le dispositif sur le client. Dans l'interface utilisateur de WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres** > **TLS (Transport Layer Security)** > **Télécharger le fichier de clés certifiées actif**. Le fichier de clés certifiées par défaut est le fichier `xsatruststore.jks`. Le mot de passe par défaut est `xc10pass`.
2. Sur la ligne de commande, définissez la variable d'environnement *`JAVA_HOME`*.
   - UNIX `export JAVA_HOME=javaHome`
   - Windows `set JAVA_HOME=javaHome`
3. Accédez au répertoire `bin`.

   ```
   cd rép_base_client_wxs/bin
   ```

4. Exécutez l'utilitaire **`xsadmin`**. Pour vous connecter au dispositif, vous devez inclure les arguments de sécurité pour le fichier de clés certifiées que vous avez téléchargé, le nom d'utilisateur et le mot de passe que vous utilisez pour vous connecter au dispositif, et le nom d'hôte de votre dispositif chaque fois que vous exécutez la commande :

   ```
   xsadmin.sh -trustPath xsatruststore.jks -trustType jks -ssl -trustPass xc10pass
   -username xcadmin -password xcadmin -ch myxc10.mycompany.com
   [additional_xsadmin_parameters]
   ```

   Vous pouvez aussi créer un fichier de configuration pour enregistrer ces paramètres. Voici un exemple de fichier de propriétés avec les paramètres requis :

   ```
   XSADMIN_TRUST_PATH=xsatruststore.jks
   XSADMIN_TRUST_TYPE=JKS
   XSADMIN_TRUST_PASS=xc10pass
   XSADMIN_USERNAME=xcadmin
   XSADMIN_PASSWORD=xcadmin
   ```

   Pour exécuter l'utilitaire **`xsadmin`** avec le fichier de propriétés, utilisez l'argument **`-profile`** pour indiquer l'emplacement du fichier de propriétés.

   ```
   xsadmin.sh -profile myxc10.properties -ssl -ch myxc10.mycompany.com
   [additional_xsadmin_parameters]
   ```

Référence de l'utilitaire xsadmin
Vous pouvez passer des arguments à l'utilitaire **`xsadmin`** selon deux méthodes différentes : avec un argument de ligne de commande ou avec un fichier de propriétés.

Migration de l'outil xsadmin vers l'outil xscmd
Dans les versions précédentes, l'outil **`xsadmin`** était un exemple d'utilitaire de ligne de commande pour surveiller l'état de l'environnement. L'outil **`xscmd`** a été introduit comme outil officiel de ligne de commande d'administration et de surveillance. Si vous utilisiez l'outil **`xsadmin`**, faites migrer les

commandes vers le nouvel outil **`xscmd`**.

Rubrique parent : Surveillance

Référence de l'utilitaire `xsadmin`

Vous pouvez passer des arguments à l'utilitaire **`xsadmin`** selon deux méthodes différentes : avec un argument de ligne de commande ou avec un fichier de propriétés.

Arguments de `xsadmin`

2.5+ **Remarque :** L'utilitaire **`xsadmin`** est obsolète. Utilisez l'utilitaire **`xscmd`** à la place. L'utilitaire **`xscmd`** est fourni comme utilitaire pris en charge pour la surveillance et l'administration de votre environnement. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Vous pouvez définir un fichier de propriétés pour l'utilitaire **`xsadmin`** avec WebSphere eXtreme Scale Client version 7.1 Correctif 1 ou ultérieur. En créant un fichier de propriétés, vous pouvez enregistrer certains des arguments fréquemment utilisés, tels que le nom d'utilisateur. Les propriétés que vous pouvez ajouter à un fichier de propriétés se trouvent dans le tableau ci-dessous. Si vous spécifiez une propriété à la fois dans un fichier de propriétés et dans l'argument de ligne de commande équivalent, la valeur de l'argument de ligne de commande remplace la valeur du fichier de propriétés.

*Tableau 1. Arguments de l'utilitaire **`xsadmin`***

Arguments de ligne de commande	Nom de la propriété équivalente dans le fichier de propriétés	Description et valeurs valides
-bp	n/a	Indique le port d'écoute. **Valeur par défaut :**`2809`
-ch	n/a	Indique le nom d'hôte JMX pour le serveur de catalogue. **Valeur par défaut :**`localhost`
-clear	n/a	Efface la mappe définie. **Permet d'utiliser les filtres suivants :** `-fm`
-containers	n/a	Pour chaque grille de données et chaque mappe définies, affiche une liste des serveurs de conteneurs. **>Permet d'utiliser les filtres suivants :** `-fnp`
-continuous	n/a	Spécifiez cet indicateur si vous voulez des résultats de taille de mappe en continu pour surveiller la grille de données. Lorsque vous exécutez cette commande avec l'argument **`-mapsizes`** , la taille de la mappe s'affiche toutes les 20 secondes.
-coregroups	n/a	Affiche tous les groupes centraux pour le serveur de catalogue. Cet argument est utilisé pour des diagnostics avancés.
-dismissLink *<domaine_service_catalogue>*	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-dmgr	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-empties	n/a	Spécifiez cet indicateur si vous voulez afficher les conteneurs vides dans les résultats.
-establishLink *<nom_domaine_étranger>*	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower

<hôte1:port1, hôte2:port2...>		XC10 Appliance.
-fc	n/a	Filtre pour ce conteneur uniquement. Si vous effectuez le filtrage des serveurs de conteneurs dans un environnement WebSphere Application Server Network Deployment, utilisez la syntaxe suivante : `<nom_cellule>/<nom_noeud>/<nomServeur_suffixeConteneur>` **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fh	n/a	Filtre pour cet hôte uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-fm	n/a	Filtre uniquement cette mappe. **Utilisez les arguments suivants : `-clear`, `-mapsizes`**
-fnp	n/a	Filtre les serveurs qui n'ont pas de fragments principaux. **Utilisez les arguments suivants : `-containers`**
-fp	n/a	Filtre pour cette partition uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-fs	n/a	Filtre pour ce serveur uniquement. Si vous effectuez le filtrage des serveurs d'applications dans un environnement WebSphere Application Server Network Deployment , utilisez la syntaxe suivante : `<nom_cellule>/<nom_noeud>/<nom_serveur>` **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fst	n/a	Filtre pour ce type de fragment uniquement. Spécifiez P pour les fragments principaux uniquement, A pour les fragments de réplique asynchrone uniquement et S pour les fragments de réplique synchrone uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`**
-fz	n/a	Filtre pour cette zone uniquement. **Utilisez les arguments suivants : `-mapsizes`, `-teardown`,`-revisions`,`-getTraceSpec`,`-setTraceSpec`,`-getStatsSpec`,`-setStatsSpec`,`-routetable`**
-force	n/a	Force l'action qui est dans la commande, en

		désactivant toutes les invites préalables. Cet argument est utile pour exécuter des commandes par lot.
-g	n/a	Spécifie le nom d'ObjectGrid.
-getstatsspec	n/a	Affiche la spécification de statistique actuelle. Vous pouvez définir la spécification de statistique avec l'argument **-setstatsspec**. **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-getTraceSpec	n/a	Affiche la spécification de trace actuelle. Vous pouvez définir la spécification de trace avec l'argument **-settracespec**. **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-h	n/a	Affiche l'aide de l'utilitaire **xsadmin** qui inclut une liste d'arguments.
-hosts	n/a	Affiche tous les hôtes de la configuration.
-jmxUrl	XSADMIN_JMX_URL	Spécifie l'adresse d'un serveur de connecteur d'API JMX au format suivant : `service:jmx:`*`protocole`*`:`*`sap`*. Les définitions des variables *`protocole`* et *`sap`* sont les suivantes : ***`protocole`*** Spécifie le protocole de transport à utiliser pour la connexion au serveur de connecteur. ***`sap`*** Spécifie l'adresse à laquelle le serveur de connecteur se trouve. Pour plus d'informations sur le format de l'URL du service JMX, voir Classe JMXServiceURL (Java™ 2 Platform SE 5.0).
-l	n/a	Affiche toutes les grilles de données et groupes de mappes connues.
-m	n/a	Spécifie le nom du groupe de mappes.
-mapsizes	n/a	Affiche la taille de chaque mappe sur le serveur de catalogue pour vérifier que la distribution des clés est uniforme sur les fragments. **Permet d'utiliser les filtres suivants :** `-fm -fst -fc -fz -fs -fh -fp`
-mbeanservers	n/a	Affiche une liste de tous les noeuds finals de serveur de bean géré.
-overridequorum	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-password	XSADMIN_PASSWORD	Spécifie le mot de passe pour se connecter à l'utilitaire **xsadmin**. Ne spécifiez pas le mot de passe dans votre fichier de propriétés si vous voulez que ce mot de passe reste sécurisé.
-p	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
	n/a	Affiche le placement configuré et le placement de l'environnement d'exécution de votre configuration. Vous pouvez définir la portée de la sortie à une combinaison de grilles de données et de groupes de

		mappes, ou à la configuration entière :
		• Configuration entière :
		`-placementStatus`
		• Pour une grille de données spécifique :
		`-placementStatus -g ma_grille`
		• Pour un groupe de mappes spécifique :
		`-placementStatus -m mon_groupe_de_mappes`
		• Pour une grille de données et un groupe de mappes spécifiques :
		`-placementStatus -g ma_grille -m mon_groupe_de_mappes`

-primaries	n/a	Affiche une liste des fragments primaires.
-profile	n/a	Spécifie un chemin complet au fichier de propriétés pour l'utilitaire **xsadmin**.
-quorumstatus	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.
-releaseShard *<nom_serveur_conteneur>* *<nom_objectgrid>* *<nom_groupe_mappes>* *<nom_partition>*	n/a	Utilisé en association avec l'argument **-reserveShard**. L'argument **-releaseShard** doit être appelé après qu'un fragment a été réservé et placé. L'argument **-releaseShard** appelle la méthode ContainerMBean.release().
-reserved	n/a	Utilisé avec l'argument **-containers** pour afficher uniquement les fragments qui ont été réservés avec l'argument **-reserveShard**.
-reserveShard *<nom_serveur_conteneur>* *<nom_objectgrid>* *<nom_groupe_mappes>* *<nom_partition>*	n/a	Transfère un fragment primaire vers le serveur de conteneur défini. La méthode ContainerMBean.reserve() est appelée par cet argument.
-resumeBalancing *<nom_objectgrid>* *<nom_ensemble_mappes>*	n/a	Tente d'équilibrer les demandes. Permet des tentatives de rééquilibrage ultérieures sur l'ObjectGrid et le groupe de mappes définis.
-revisions	n/a	Affiche les identificateurs des révisions d'un domaine de service de catalogue avec chaque grille de données, le numéro de partition, le type de partition (principale ou réplique), le domaine de service de catalogue, l'ID de durée de vie et le nombre de révisions des données de chaque fragment. Vous pouvez utiliser cet argument pour déterminer si une réplique asynchrone ou un domaine lié sont interceptés. Cet argument appelle

		la méthode ObjectGridMBean.getKnownRevisions(). **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-routetable	n/a	Affiche l'état actuel de la grille de données à partir d'une perspective serveur client. La table de routage est l'information qu'un serveur client ObjectGrid utilise pour communiquer avec la grille de données. Utilisez la table de routage sous la forme d'une aide au diagnostic lorsque vous tentez d'identifier les problèmes de connexion ou les exceptions `TargetNotAvailable`. **Arguments requis :** Dans un environnement autonome, vous devez spécifier les paramètres **-bp** et **-p** les avec cet argument si vous n'utilisez pas les valeurs par défaut pour le port d'écoute d'amorçage et le port JMX pour l'hôte du serveur de catalogue. **Permet d'utiliser les filtres suivants :** `-fz -fh -fp`
-settracespec *<chaîne_trace>*	n/a	Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance. Voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance pour plus d'informations sur la configuration de la trace sur le dispositif. **Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp`
-swapShardWithPrimary *<nom_serveur_conteneur>* *<nom_objectgrid>* *<nom_ensemble_mappes>* *<nom_partition>*	n/a	Permute le fragment de réplique défini du serveur de conteneur indiqué et le fragment primaire. Cette commande permet d'équilibrer manuellement les fragments primaires lorsque cela est nécessaire.
-setstatsspec *<spécification_stats>*	n/a	Active la collecte des statistiques. Cet argument appelle les méthodes DynamicServerMBean.setStatsSpec et DynamicServerMBean.getStatsSpec. **Permet d'utiliser les filtres suivants :** `-fm -fst -fc -fz -fs -fh -fp`
-suspendBalancing *<nom_objectgrid>* *<nom_ensemble_mappes>*	n/a	Empêche les tentatives d'équilibrage de l'ObjectGrid et du groupe de mappes définis.
-ssl	n/a	Indique que SSL (Secure Sockets Layer) est activé.
-teardown	n/a	**Permet d'utiliser les filtres suivants :** `-fst -fc -fz -fs -fh -fp` **Format pour spécifier une liste de serveurs :**

		Format pour spécifier une liste de serveurs :
		`nom_serveur_1,nom_serveur_2 ...`
		Pour arrêter tous les serveurs dans une zone, incluez l'argument -fz :
		`–fz <nom_zone>`
		Pour arrêter tous les serveurs sur un hôte, incluez l'argument -fh :
		`–fh <nom_hôte>`
		Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.

	n/a	Force le placement de fragment à s'exécuter, en ignorant la valeur **numInitialContainers** définie dans le fichier de déploiement XML. Vous pouvez utiliser cet argument lorsque vous effectuez des opérations de maintenance pour pouvoir continuer à placer les fragments, même si la valeur **numInitialContainers** est inférieure à la valeur définie.
-trustPass	XSADMIN_TRUST_PASS	Spécifie le mot de passe pour le fichier de clés certifiées spécifié.
-trustPath	XSADMIN_TRUST_PATH	Spécifie un chemin vers le fichier de clés certifiées. Exemple : `etc/test/security/server.public`
-trustType	XSADMIN_TRUST_TYPE	Spécifie le type de fichier de clés certifiées. Les valeurs valides sont : JKS, JCEK, PKCS12, etc.
-unassigned	n/a	Affiche une liste de fragments qui ne peuvent pas être placés sur la grille de données. Les fragments ne peuvent pas être placés lorsque le service de placement a une contrainte qui empêche le placement.
-username	XSADMIN_USERNAME	Spécifie le nom d'utilisateur pour se connecter à l'utilitaire **xsadmin**.
-v	n/a	Active l'action de ligne de commande prolixe. Utilisez cet indicateur si vous utilisez des variables d'environnement, un fichier de propriétés, ou les deux pour spécifier certains arguments de ligne de commande, et si vous voulez afficher leur valeur.
-xml	n/a	Affiche la sortie filtrée à partir de la méthode PlacementServiceMBean.listObjectGridPlacement(). Les autres arguments **xsadmin** filtrent la sortie de cette méthode et organisent les données dans un format plus exploitable.

Rubrique parent : Surveillance avec l'utilitaire xsadmin

Migration de l'outil `xsadmin` vers l'outil `xscmd`

Dans les versions précédentes, l'outil **`xsadmin`** était un exemple d'utilitaire de ligne de commande pour surveiller l'état de l'environnement. L'outil **`xscmd`** a été introduit comme outil officiel de ligne de commande d'administration et de surveillance. Si vous utilisiez l'outil **`xsadmin`**, faites migrer les commandes vers le nouvel outil **`xscmd`**.

Equivalents des commandes `xsadmin` et `xscmd`

2.5+ **Important :** L'utilitaire **`xsadmin`** est obsolète. Utilisez l'utilitaire **`xscmd`** à la place. L'utilitaire **`xscmd`** est fourni comme utilitaire pris en charge pour la surveillance et l'administration de votre environnement. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

*Tableau 1. Arguments de l'utilitaire **`xsadmin`** et commandes équivalentes **`xscmd`**. Certaines commandes **`xscmd`** ont une f forme longue. La forme courte des commandes a un tiret (-) et la forme longue, deux (--). Vous pouvez utiliser l'une à la inversement.*

Arguments de ligne de commande `xsadmin`	**Commande équivalente `xscmd`**	**Paramètres de commande `xscmd`**
`-bp`	• **`-cep`** *hostname*:*listener_port* • **`--catalogEndpoint`** *hostname*:*listener_port*	n/a
`-ch`	• **`-cep`** *hostname*:*listener_port* • **`--catalogEndpoint`** *hostname*:*listener_port*	n/a
`-clear`	**`-c clearGrid`**	**`-g, -ms, -v, -m, (-cep)`**
`-containers`	• **`-c showPlacement -container`** *containerName* • **`-c showPlacement -server`** *serverName*	**`-e, -i, , -st, -snp, -ct, -s, -p, -hf`**
`-continuous`	n/a	n/a
`-coregroups`	• • **`-c listCoreGroupMembers -cg`** *core_group*	n/a
`-dismissLink` *<catalog_service_domain>*	**`-c dismissLink`**	• **`-fd`** *<foreignCatalogServiceDomain* • **`--foreignCatalogServiceDomain`** *<foreignCatalogServiceDomain>*
`-dmgr`	s/o - Cet argument est déterminé automatiquement avec **`xscmd`**	n/a
`-empties`	arg spécifique d'une nouvelle commande	n/a
`-establishLink` *<foreign_domain_name> <host1:port1,host2:port2...>*	**`-c establishLink`**	• **`-fd`** *<foreignCatalogServiceDomain <host1:port1,host2:port2...>* • **`--foreignCatalogServiceDomain`** *<foreignCatalogServiceDomain>* **`-1`** *<host1:port1,host2:port2...>*
`-fc`	• **`-ct`** • **`--container`**	n/a
`-fh`	• **`-hf`** • **`--hostFilter`**	n/a
`-fm`	• **`-m`** • **`--map`**	n/a
`-fnp`	• **`-snp`** • **`--serversWithNoPrimaries`**	n/a
`-fp`	• **`-p`** • **`--partitionId`**	n/a

-fs	• **-s** • **--server**	n/a
-fst	• **-st** <*shard_type*> • **--shardType** <*shard_type*> Shard values: P=primary A=asyncReplica S=syncReplica	n/a
-fz	• **-z** • **--zone**	n/a
-force	arg spécifique d'une nouvelle commande	
-g	• **-g** • **--objectGrid**	n/a
-getstatsspec	**-c getStatsSpec**	n/a
-getTraceSpec	**-c getTraceSpec**	n/a
-h	Vous pouvez exécuter l'aide avec ou sans un nom de commande spécifique : • **-h** • **--help** • **-h** <*command_name*> • **--help** <*command_name*>	n/a
-hosts	**-c listHosts**	**-g, -ms, -st, -c, -s, -hf, -z**
-jmxUrl	• **-cep** *hostname*:*listener_port* • **--catalogEndpoint** *hostname*:*listener_port*	n/a
-l	**-c listObjectGridNames**	n/a
-m	• **-ms** • **--mapSet**	n/a
-mapsizes	**-c showMapSizes**	**-g, -ms, -i, [-ct, -z, -s, -hf, sht [**
-mbeanservers	**-c listAllJMXAddresses**	n/a
-overridequorum	**-c overrideQuorum**	n/a
-password	• **-pwd** • **--password**	n/a
-p	• **-cep** *hostname*:*listener_port* • **--catalogEndpoint** *hostname*:*listener_port*	n/a
-placementStatus	**-c placementServiceStatus**	**-g, -ms**
-primaries	**-c showPlacement -sf P**	**-e, -i, , -st, -snp, -ct, -s, -p, -hf**
-profile	Pour enregistrer les paramètres de sécurité actuels dans un profil de sécurité : • **-ssp** *profile_name* • **--saveSecProfile** *profile_name* Pour utiliser un profil de sécurité spécifié : • **-sp** *profile_name* • **--securityProfile** *profile_name*	
-quorumstatus	**-c showQuorumStatus**	n/a
-releaseShard	**-c releaseShard**	**-c, -g, -ms, -p**

Avertissement : Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	
-reserved	• **-sf R** • **--shardFilter R**	n/a
-reserveShard **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c reserveShard** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c, -g, -ms, -p**
-resumeBalancing *<objectgrid_name>* *<map_set_name>* **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c resumeBalancing** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-g, -ms**
-revisions	**-c revisions**	**-s,-p,-g,-m**
-routetable	**-c routetable**	**-z, -hf,-p,-g,-ms**
-settracespec *<trace_string>*	**-c setTraceSpec**	**-spec** *<trace_string>*
-swapShardWithPrimary **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c swapShardWithPrimary** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c -g, -ms, -p**
-setstatsspec *<stats_spec>*	**-c setStatsSpec**	**-spec** *<stats_spec>*
-suspendBalancing *<objectgrid_name>* *<map_set_name>* **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c suspendBalancing** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-g, -ms**
-ssl	• **-ssl** • **--enableSSL**	n/a
-teardown **Avertissement :** Cet argument de ligne de	**-c teardown** **Avertissement :** Cet argument de ligne de commande ne s'applique pas	**-f, , -st, -snp, -c, -s, -p, -hf, -z,**

commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	pour les configurations WebSphere DataPower XC10 Appliance.	
-triggerPlacement **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-c triggerPlacement** **Avertissement :** Cet argument de ligne de commande ne s'applique pas pour les configurations WebSphere DataPower XC10 Appliance.	**-g, -ms**
-trustPass	• **-tsp** • **--trustStorePassword**	n/a
-trustPath	• **-ts** • **--trustStore**	n/a
-trustType	• **-tst** • **--trustStoreType**	n/a
-unassigned	**-c showPlacement -sf U**	**-e, -i, , -st, -snp, -ct, -s, -p, -hf**
-username	• **-user** • **--username**	n/a
-v	• **-v** • **--verbose**	n/a
-xml	**-c showPlacement**	n/a

Rubrique parent : Surveillance avec l'utilitaire xsadmin

Surveillance de la santé de l'environnement

2.5+ Message Center fournit une vue agrégée des notifications d'événements pour les messages de journal et de l'outil de diagnostic de premier niveau. Vous pouvez afficher ces notifications d'événements avec : Message Center dans la console Web, l'utilitaire **xscmd**, les fichiers journaux de santé, l'interface de commande HTTP, ou un programme avec MBeans.

2.5+ **Surveillance de l'état de santé du système - présentation**
Message Center agrège en temps réel les événements d'état de santé provenant de tous les serveurs de conteneur et de catalogue d'une collectivité. Lorsque Message Center est configuré, vous pouvez afficher une présentation des événements critiques qui surviennent sur différents serveurs sans collecter les journaux de chaque serveur.

2.5+ **Affichage des notifications d'événement de santé dans Message Center**
Vous pouvez utiliser le centre de messages (Message Center) de la console Web pour évaluer en temps réel la santé de l'intégralité de la grille de données et de la collectivité. Les événements qui s'affichent dans Message Center sont un sous-ensemble d'événements qui sont filtrés pour afficher les problèmes les plus critiques.

2.5+ **Affichage des informations de santé avec l'utilitaire xscmd**
Vous pouvez afficher les notifications d'événements en cours, l'historique de notification d'événement et afficher et définir des filtres de notification à partir du centre de message (Message Center) avec l'utilitaire **xscmd**.

Que faire ensuite

Vous pouvez afficher un récapitulatif de l'état de santé global du matériel et des logiciels du dispositif. Utilisez pour cela l'une des options suivantes :

- Affichez et téléchargez le fichier journal de santé. Pour plus d'informations, voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance.
- Utilisez la commande **GetHealthStatus** dans l'interface de commande HTTP. Pour plus d'informations, voir Administration à l'aide de l'interface de commande HTTP et Référence de l'interface de commande HTTP.

Rubrique parent : Surveillance

Surveillance de l'état de santé du système - présentation

2.5+ Message Center agrège en temps réel les événements d'état de santé provenant de tous les serveurs de conteneur et de catalogue d'une collectivité. Lorsque Message Center est configuré, vous pouvez afficher une présentation des événements critiques qui surviennent sur différents serveurs sans collecter les journaux de chaque serveur.

Mise en oeuvre de Message Center

Le centre de messages Message Center est activé par défaut. Vous pouvez le désactiver dans l'interface utilisateur.

Les déploiements de grille de données peuvent impliquer des dizaines, voire des centaines, de processus serveur répartis. Si un incident se produit, vous pouvez ouvrir le fichier journal du serveur de conteneur concerné pour analyser le problème en détail.

Message Center est constitué des composants suivants :

Agrégation d'événements

Lorsque vous configurez la surveillance de la santé sur un serveur de catalogue, vous recevez les événements agrégés qui affectent la santé de l'ensemble du domaine de service de catalogue. Les informations reçues incluent la source et la gravité des types d'événement suivants :

- Tous les événements de l'outil de diagnostic de premier niveau
- Toutes les entrées de journal WARNING ou SEVERE
- La liste de toutes les entrées de journal, notamment INFO, WARNING et SEVERE
- Les opérations de démarrage et d'arrêt du serveur
- La grille de données approche de sa limite de capacité
- Perte ou récupération du quorum
- Déclenchement des alertes SNMP activées
- Retard dans la réplication pendant une période de 15 minutes

Message Center dans la console Web

Message Center dans la console Web affiche les enregistrements d'événement agrégés. Ces événements incluent les événements récents et les notifications de mise à jour en temps réel qui se produisent après l'ouverture de la console.

Evénements dans l'utilitaire `xscmd`

Vous pouvez également afficher la liste des événements récents à l'aide de l'utilitaire **`xscmd`**. Lorsque des événements se produisent, vous pouvez rediriger les enregistrements d'événement pour créer des utilitaires de scriptage automatique.

Beans géré pour l'intégration dans d'autres logiciels de surveillance

Vous pouvez également utiliser les beans gérés de gestion (MBeans) disponibles pour connecter Message Center à vos autres logiciels de surveillance JMX (Java Management Extensions). La documentation de ces beans gérés est incluse dans la documentation des API.

2.5+

Récapitulatif de l'état de santé

Les journaux de santé et la commande **`GetHealthStatus`** de l'interface de commande HTTP vous permettent d'obtenir un récapitulatif de l'état de santé du matériel et des logiciels du dispositif. Les informations fournies incluent le placement de la grille de données et son état de réplication, des avertissements concernant le matériel, l'état de santé du système et du réseau, et d'autres messages d'état.

Message Center et Analyseur de journal

L'analyseur de journal est un autre outil qui permet d'analyser un groupe de messages de journal. Cet outil impose de collecter manuellement les journaux des divers serveurs dans votre environnement. Ensuite, vous pouvez exécuter l'outil pour créer des rapports sur les problèmes. Utilisez l'analyseur de journal pour exécuter une analyse post-mortem des journaux lorsque vous devez analyser un nombre de messages supérieur au sous-ensemble de 1000 messages que vous pouvez afficher dans Message Center. Utilisez Message Center pour surveiller en temps réel l'état de la grille de données pour identifier rapidement les problèmes qui surviennent. Ensuite, vous pouvez vérifier les fichiers journaux du serveur de conteneur associé ou utiliser l'analyseur de journal pour analyser le problème plus en détail.

Configuration et architecture de la surveillance de la santé

Message Center est activé par défaut. Vous pouvez le désactiver dans l'interface utilisateur. Lorsque Message Center est activé, les serveurs de catalogue de votre collectivité jouent le rôle de concentrateurs pour la surveillance de l'état de santé. Généralement, les messages de Message Center proviennent du

serveur de catalogue qui réside sur le dispositif à partir duquel vous avez créé la collectivité. Chaque concentrateur dispose de ses propres abonnements et historiques d'événements. Chaque événement de l'historique porte un numéro de séquence. Les historiques d'événements sur les serveurs de catalogue ne sont pas synchronisés et sont différents. Les serveurs de catalogue peuvent s'abonner aux événements de journal et de l'outil de diagnostic de premier niveau des autres serveurs de catalogue.

Rubrique parent : 2.5+ Surveillance de la santé de l'environnement

Affichage des notifications d'événement de santé dans Message Center

Vous pouvez utiliser le centre de messages (Message Center) de la console Web pour évaluer en temps réel la santé de l'intégralité de la grille de données et de la collectivité. Les événements qui s'affichent dans Message Center sont un sous-ensemble d'événements qui sont filtrés pour afficher les problèmes les plus critiques.

Procédure

- Affichez les erreurs graves, les messages de l'outil de diagnostic de premier niveau et démarrez et arrêtez les événements serveur via les notifications d'événements dans la console Web. Ces notifications s'affichent automatiquement lorsque vous êtes connecté à une page dans la console Web.
- Affichez les messages dans Message Center. Dans la console Web, cliquez sur **Surveillance** > **Message Center**. Message Center affiche les 1000 derniers messages critiques qui ont été envoyés via le concentrateur de messages du serveur de catalogue.

  Le concentrateur de messages du serveur de catalogue filtre les messages qui s'affichent et il affiche donc les 1000 messages. Par conséquent, Message Center contient un sous-ensemble de tous les événements critiques qui surviennent sur les serveurs de catalogue. Si de nouveaux messages deviennent disponibles alors que la page est ouverte, un message d'information comportant une option d'actualisation des informations s'affiche en haut de la page.
- Filtrez les messages affichés dans Message Center. Vous pouvez ajouter jusqu'à trois règles de filtrage. Une règle est constituée d'une colonne, d'une condition et d'une valeur.
  1. Dans la console Web, cliquez sur **Surveillance** > **Message Center**.
  2. Cliquez sur le bouton de filtrage ( ).
  3. Ajoutez une règle.
     a. Cliquez sur le bouton d'ajout ( ).
     b. Dans Message Center, choisissez la colonne à filtrer :

        ID

        ID d'événement généré par Message Center.

        Type

        Le type de message qui indique la gravité du message. Les valeurs admises sont : Grave, Avertissement, Erreur et Information.

        Date

        Date et heure de génération du message.

        Source

        Serveur d'origine du message.

        Message

        Texte du message de l'événement de message.

     c. Sélectionnez la condition à laquelle vous voulez appliquer le filtre. La liste suivante de conditions s'appliquent à pratiquement toutes les colonnes, à l'exception de la colonne de date et de type :
        - Contient
        - Est
        - Commence par
        - Se termine par

     d. Entrez la valeur de filtrage de la colonne.

     Exemple : pour n'afficher que les messages de server1, sélectionnez la colonne **Source**. Sélectionnez la condition **Est**. Pour la valeur, tapez `server1`.
  4. Vous pouvez choisir d'établir une correspondance avec certaines des règles que vous avez définies ou toutes les règles.
  5. Cliquez sur **Filtrer** pour appliquer les filtres définis à la sortie de Message Center.
- 2.5+ Désactivez Message Center. La désactivation de Message Center interrompt la détection de nouveaux messages. Cliquez sur **Désactiver Message Center et les alertes en incrustation**. Cela fait, appliquez la modification.

Que faire ensuite

Vous pouvez exécuter une analyse supplémentaire sur vos fichiers journaux. Pour plus d'informations, voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance.

Rubrique parent : 2.5+ Surveillance de la santé de l'environnement

Affichage des informations de santé avec l'utilitaire `xscmd`

2.5+ Vous pouvez afficher les notifications d'événements en cours, l'historique de notification d'événement et afficher et définir des filtres de notification à partir du centre de message (Message Center) avec l'utilitaire **`xscmd`**.

Avant de commencer

- Démarrez l'utilitaire **`xscmd`** et connectez-le au domaine de service de catalogue. Pour plus d'informations, voir Administration avec l'utilitaire xscmd.

Procédure

- Affichez l'historique des notifications d'événements avec l'utilitaire **`xscmd`**. La sortie apparaît dans un tableau.

```
xscmd -c showNotificationHistory -cep nom_hôte:port(,nom_hôte:port)
```

- Ecoutez les nouvelles notifications.

```
xscmd -c listenForNotifications -cep nom_hôte:port(,nom_hôte:port)
```

  La sortie apparaît dans un format brut et s'exécute jusqu'à ce que vous arrêtiez la commande. Vous pouvez écrire des scripts supplémentaires pour analyser la sortie.

- Activez le filtrage pour toutes les entrées de journal à venir, y compris INFO, WARNING et SEVERE. Par défaut, Message Center et les commandes affichent uniquement les erreurs WARNING et SEVERE et les événements. Vous pouvez définir le filtre pour tous les serveurs dans l'environnement ou sur un serveur unique. Ce paramétrage n'affecte que les résultats à venir.

```
xscmd -c setNotificationFilter -fs <expression régulière>  [-server <nom_serveur>]
```

- Affichez les filtres de notification en cours pour tous les serveurs de l'environnement ou un serveur unique.

```
xscmd -c getNotificationFilter [-s nom_serveur]
```

Rubrique parent : 2.5+ Surveillance de la santé de l'environnement

Surveillance avec SNMP (Simple Network Monitoring Protocol)

Avec le support SNMP (Simple Network Monitoring Protocol), vous pouvez contrôler l'état d'un dispositif IBM® WebSphere DataPower XC10 Appliance dans le cadre d'un grand groupe de systèmes dans un centre de données. La surveillance SNMP améliore votre capacité de notifier les problèmes et de les résoudre rapidement.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez utiliser le contrôle que si un client est configuré.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Pourquoi et quand exécuter cette tâche

Procédez comme suit pour activer la surveillance SNMP pour votre IBM WebSphere DataPower XC10 Appliance. Les paramètres SNMP sont uniques pour le dispositif et ne sont pas propagés auprès des autres dispositifs de la collectivité.

A faire : Les statistiques SNMP étant fournies pour un seul dispositif, les valeurs rapportées sont différentes des valeurs qui sont rapportées par les panneaux de surveillance dans l'interface utilisateur. L'interface utilisateur rapporte des données sur la collectivité. Par exemple, une valeur de la table gridStats, de la table mapStats ou de la table jvmStats SNMP représente un sous-ensemble de la valeur totale rapportée dans l'interface utilisateur.

1. Activation de la surveillance SNMP du dispositif
   La surveillance SNMP (Simple Network Monitoring Protocol) peut être activée pour le dispositif IBM WebSphere DataPower XC10 Appliance. Vous pouvez télécharger les bases d'informations de gestion (MIB) fournies dans le dispositif pour définir les données SNMP disponibles pour le client SNMP.
2. Configuration de communautés SNMP
   Vous pouvez définir l'accès aux données SNMP (Simple Network Management Protocol) sur le dispositif en créant une ou plusieurs communautés SNMP. Une communauté SNMP est nécessaire lorsque le contrôle est activé.
3. Configuration des abonnés aux interceptions SNMP
   Les abonnés aux interceptions représentent les clients SNMP (Network Monitoring Protocol) utilisés pour communiquer avec l'agent SNMP intégré au dispositif.
4. Gestion des abonnements aux interceptions SNMP
   Une interruption SNMP (Simple Network Monitoring Protocol) est une notification d'événement ou d'état générée par l'agent SNMP intégré au dispositif. En utilisant des abonnements aux interceptions SNMP, vous définissez les interceptions SNMP que l'agent communique aux clients SNMP.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Rubrique parent : Surveillance

Activation de la surveillance SNMP du dispositif

La surveillance SNMP (Simple Network Monitoring Protocol) peut être activée pour le dispositif IBM® WebSphere DataPower XC10 Appliance. Vous pouvez télécharger les bases d'informations de gestion (MIB) fournies dans le dispositif pour définir les données SNMP disponibles pour le client SNMP.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez utiliser le contrôle que si un client est configuré.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Pourquoi et quand exécuter cette tâche

Par défaut, la surveillance SNMP n'est pas activée pour IBM WebSphere DataPower XC10 Appliance. Les paramètres SNMP sont uniques pour le dispositif et ne sont pas propagés auprès des autres dispositifs de la collectivité.

Procédure

1. Accédez au panneau de surveillance. Dans l'interface utilisateur de IBM WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres SNMP**.
2. Cochez la case pour activer la surveillance SNMP. Si vous voulez la désactiver, désélectionnez la case.
3. Téléchargez les fichiers MIB (Management Information Base) qui sont disponibles sur le dispositif.

   Les fichiers Enterprise MIB décrivent les fonctions et les données disponibles auprès de l'agent SNMP imbriqué pour que votre client puisse y accéder de façon appropriée. Le client peut lancer les commandes SNMP `GET`, `GET-NEXT` et `GET-BULK`. Vous pouvez télécharger ces fichiers MIB et les importer vers le client pour accéder aux données en plus des définitions de données MIB-II. Développez les **MIB d'entreprise** et cliquez sur le nom de chaque base d'informations de gestion à partir du dispositif.

   Statistiques MIB

   Les statistiques MIB comprennent des informations similaires aux statistiques que vous pouvez voir avec la fonction de surveillance de l'interface utilisateur. La base d'informations de gestion MIB contient aussi des statistiques sur la machine virtuelle Java™.

   MIB statut matériel

   Le MIB statut matériel comprend des informations sur le statut du matériel : les températures, la date et l'heure, etc.

   MIB notifications matériel

   La **`MIB notifications matériel`** permet de définir les informations disponibles pour les commandes `SNMP TRAP`.

Résultats

A l'issue de ces étapes, vous avez activé la surveillance SNMP pour le dispositif et téléchargés les données MIB pour le client SNMP.

Que faire ensuite

Utilisez votre client SNMP pour afficher les données MIB.

A faire : Les statistiques SNMP étant fournies pour un seul dispositif, les valeurs rapportées sont différentes des valeurs qui sont rapportées par les panneaux de surveillance dans l'interface utilisateur. L'interface utilisateur rapporte des données sur la collectivité. Par exemple, une valeur de la table gridStats, de la table mapStats ou de la table jvmStats des statistiques SNMP est un sous-ensemble de la valeur totale rapportée dans l'interface utilisateur.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique suivante : Configuration de communautés SNMP

Configuration de communautés SNMP

Vous pouvez définir l'accès aux données SNMP (Simple Network Management Protocol) sur le dispositif en créant une ou plusieurs communautés SNMP. Une communauté SNMP est nécessaire lorsque le contrôle est activé.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez utiliser le contrôle que si un client est configuré. WebSphere DataPower XC10 Appliance prend en charge SNMP version v2c.

Pourquoi et quand exécuter cette tâche

Une communauté est nécessaire pour s'authentifier auprès de l'agent SNMP intégré et accéder aux données SNMP. L'agent SNMP intégré au dispositif attend que le client fournisse un nom de communauté défini pour s'authentifier auprès de l'agent et retourner les données demandées. Si aucune communauté n'est fournie, l'agent SNMP ignore la demande du client. Les communautés SNMP sont uniques pour le dispositif et ne sont pas propagées auprès des autres dispositifs de la collectivité.

Procédure

1. Accédez au panneau de surveillance. Dans la barre de menus dans la partie supérieure de l'interface utilisateur de IBM® WebSphere DataPower XC10 Appliance, accédez à **Dispositif** > **Surveillance SNMP**.
2. Développez **Communautés SNMP**.
3. Cliquez sur **Créer une communauté**.
4. Complétez le formulaire d'abonnement à la communauté SNMP à créer.

   `Nom`

   Nom décrivant une communauté SNMP.

   `Restriction hôte`

   Cette zone spécifie une adresse IP à utiliser pour la communication aux formats IPv4 et IPv6. Elle accepte également les noms d'hôte résolus en hôte. Vous pouvez limiter ultérieurement l'accès à un sous-réseau spécifié à l'aide d'une adresse IP ou d'un nom d'hôte de routage CIDR (Classless Inter-Domain Routing). Si une restriction d'hôte est incluse, les communications avec une adresse adresse IP ou un autre sous-réseau sont refusées. Si vous laissez cette zone vide, vous autorisez la communication avec toutes les adresses IP.

5. Cliquez sur l'icône de suppression ( ) pour supprimer une communauté SNMP. Vous ne pouvez pas modifier les communautés existantes. Si vous devez modifier une communauté, vous devez la supprimer et la recréer.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique précédente : Activation de la surveillance SNMP du dispositif
Rubrique suivante : Configuration des abonnés aux interceptions SNMP

Configuration des abonnés aux interceptions SNMP

Les abonnés aux interceptions représentent les clients SNMP (Network Monitoring Protocol) utilisés pour communiquer avec l'agent SNMP intégré au dispositif.

Avant de commencer

Les abonnés aux interceptions SNMP doivent être des clients SNMP existants. Configurez le client SNMP pour recevoir des interceptions avant de créer l'abonné aux interceptions dans le dispositif IBM® WebSphere DataPower XC10 Appliance interface utilisateur.

Pourquoi et quand exécuter cette tâche

Un abonné aux interceptions SNMP est requis pour l'acheminement des notifications relatives au dispositif à un client SNMP abonné. Lors de la création d'un abonné aux interceptions, vous devez fournir des informations sur les clients SNMP qui reçoivent des interceptions SNMP. Les interceptions décrivent les situations nécessitant l'attention de l'administrateur d'IBM WebSphere DataPower XC10 Appliance. L'agent SNMP n'envoie pas d'interceptions aux clients qui ne sont pas définis comme abonnés.

Tous les abonnés aux interceptions SNMP reçoivent automatiquement des interceptions relatives aux limites de stockage de la grille de données et aux limites de stockage du dispositif. Pour plus d'informations sur les interceptions spécifiques pouvant être définies, voir Référence d'interruption SNMP.

Procédure

1. Accédez au panneau Paramètres SNMP. Dans l'interface utilisateur de WebSphere DataPower XC10 Appliance, cliquez sur **Dispositif** > **Paramètres SNMP**.
2. Développez **Abonnés aux interceptions**.
3. Cliquez sur **Créer un abonné aux interceptions**.
4. Complétez le formulaire pour décrire les abonnés aux interceptions SNMP à créer.

   `Hôte`

   Définit l'adresse IP sur laquelle le client SNMP écoute les informations d'interception.

   `Numéro de port du client`

   Définit le port sur lequel le client SNMP écoute les informations d'interception.

   `Communauté`

   Définit une communauté SNMP dont le client est membre.

5. Cliquez sur l'icône de suppression ( ) pour supprimer un abonné aux interceptions SNMP. Vous ne pouvez pas modifier les abonnés existants. Si vous devez modifier un abonné, vous devez le supprimer et le recréer.

Que faire ensuite

Après avoir défini au moins un abonné aux interceptions SNMP qui peut accéder au dispositif, vous pouvez configurer les interceptions que doit communiquer l'agent SNMP.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique précédente : Configuration de communautés SNMP
Rubrique suivante : Gestion des abonnements aux interceptions SNMP

Gestion des abonnements aux interceptions SNMP

Une interruption SNMP (Simple Network Monitoring Protocol) est une notification d'événement ou d'état générée par l'agent SNMP intégré au dispositif. En utilisant des abonnements aux interceptions SNMP, vous définissez les interceptions SNMP que l'agent communique aux clients SNMP.WebSphere DataPower XC10 Appliance prend en charge SNMP version 2vc.

Avant de commencer

Configurez un client SNMP avant d'exécuter ces étapes. Vous pouvez exécuter ces étapes sans client SNMP configuré, mais vous ne pouvez pas recevoir les interceptions SNMP sans client SNMP valide configuré.

Pourquoi et quand exécuter cette tâche

Pour définir les interceptions à communiquer, procédez comme suit.

Procédure

1. Accédez au panneau Paramètres SNMP. Dans la barre de menus dans la partie supérieure de l'interface utilisateur de WebSphere DataPower XC10 Appliance, accédez à **Dispositif** > **Paramètres SNMP** > **Abonnements aux interceptions**.
2. Sélectionnez ou désélectionnez des interceptions individuelles. Vous pouvez sélectionnez les interceptions SNMP en sélectionnant et en effaçant des interceptions individuelles. Tous les abonnés aux interceptions SNMP reçoivent automatiquement des interceptions relatives aux limites de stockage de la grille de données et aux limites de stockage du dispositif. Pour plus d'informations sur les interceptions spécifiques pouvant être définies, voir Référence d'interruption SNMP.

Résultats

A l'issue de ces étapes, vous avez défini le groupe d'interceptions SNMP signalées aux clients SNMP abonnés.

Référence d'interruption SNMP
Les événements ci-dessous peuvent être interceptés par l'agent SNMP (IBM WebSphere DataPower XC10 Appliance Simple Network Management Protocol) et communiqués aux clients SNMP abonnés.

Rubrique parent : Surveillance avec SNMP (Simple Network Monitoring Protocol)
Rubrique précédente : Configuration des abonnés aux interceptions SNMP

Référence d'interruption SNMP

Les événements ci-dessous peuvent être interceptés par l'agent SNMP (IBM® WebSphere DataPower XC10 Appliance Simple Network Management Protocol) et communiqués aux clients SNMP abonnés.

Interceptions SNMP disponibles pour contrôler votre dispositif IBM WebSphere DataPower XC10 Appliance

Tableau 1. Interceptions SNMP de IBM WebSphere DataPower XC10 Appliance.

Les interceptions ci-dessous sont configurées par défaut. Vous ne pouvez pas les désactiver.

Description
Cette notification indique que la grille de données XC10 spécifique a atteint 60 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifiée est repassée en dessous de 60 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifique a atteint 80 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifiée est repassée en dessous de 80 % de sa capacité.
Cette notification indique que la grille de données XC10 spécifique a atteint sa capacité.
Cette notification indique que la grille de données XC10 spécifiée est repassée en dessous de sa capacité.
Cette notification indique que le stockage du dispositif XC10 a atteint 80 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 est repassé en dessous de 80 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 a atteint 90 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 est repassé en dessous de 90 % de sa capacité.
Cette notification indique que le stockage du dispositif XC10 a atteint sa capacité.
Cette notification indique que le stockage du dispositif XC10 est repassé en dessous de sa capacité.

Tableau 2. Interceptions SNMP du dispositif.

Les interceptions ci-dessous sont activées par défaut. Vous pouvez les désactiver dans l'interface utilisateur.

Valeur par défaut	Description
Y	L'interface Ethernet passe à l'état inactif.
Y	L'interface Ethernet est passée à l'état actif.
Y	L'agent SNMP est réinitialisé avec des modifications de configuration potentielles
Y	L'agent SNMP est en cours d'arrêt.
Y	Le capteur du ventilateur est repassé à l'état normal.
Y	Le capteur du ventilateur est passé à un état indésirable.
Y	Le détecteur de tension est repassé à l'état normal.
Y	Le capteur de tension est passé à un état indésirable.
Y	Le détecteur de température est repassé à une plage normale.
Y	Le détecteur de température est passé à une plage indésirable.
Y	Le capteur d'alimentation est repassé à l'état normal.

Y	Le capteur d'alimentation est repassé à l'état normal.
Y	Le capteur d'alimentation est passé à un état indésirable.

Rubrique parent : Gestion des abonnements aux interceptions SNMP

Configuration de la consignation à distance

Vous pouvez activer la consignation à distance pour enregistrer les entrées de journal sur un serveur distant en dehors du dispositif. La consignation à distance peut être utile lorsque vous devez définir un niveau de journal de débogage détaillé pour isoler un problème ou surveiller un comportement sur une longue période.

Avant de commencer

- Vous devez disposer d'un serveur syslog qui écoute les événements et les capture.
- Les noms des serveurs de catalogue, de conteneur et d'applications (si vous utilisez WebSphere Application Server) ne doivent contenir que des caractères alphanumériques. Syslog RFC 1364 n'autorise pas les caractères non alphanumériques pour la zone TAG. La zone TAG contient le nom du serveur dans les messages syslog.

Pourquoi et quand exécuter cette tâche

Utilisez la consignation à distance pour analyser les données d'historique. Le dispositif conserve un nombre limité de fichiers journaux dans le système. Configurez la consignation à distance si vous voulez enregistrer un plus grand nombre de fichiers journaux pour l'analyse. Le serveur de consignation à distance agrège les données de plusieurs dispositifs afin que vous puissiez configurer l'ensemble de la collectivité pour envoyer des fichiers au même serveur de consignation à distance.

Procédure

Configurez un serveur de consignation à distance dans le dispositif interface utilisateur.

a. Dans interface utilisateur, cliquez sur **Dispositif** > **Identification des incidents** > **Journaliation** > **Configurer la consignation à distance**.
b. Entrez l'hôte distant et le port du serveur de journalisation distant.
c. Entrez le seuil de gravité des messages à envoyer au serveur de journalisation distant. Pour envoyer des messages d'avertissement et de gravité, sélectionnez **WARNING**. Pour envoyer des messages graves, sélectionnez **SEVERE**.

Résultats

Les messages sont envoyés au serveur de journalisation distant configuré pour l'archivage et l'analyse.

Rubrique parent : Surveillance

Java

Configuration d'une application de grille de données pour l'utilisation de l'authentification client

Si la grille de données que vous configurez pour l'application utilise la sécurité, vous devez configurer un fichier `client.properties` comportant des paramètres à transmettre à l'application de grille de données.

Avant de commencer

Vous devez disposer d'une application de grille de données qui utilise WebSphere DataPower XC10 Appliance.

Procédure

1. Sécurisez la grille de données utilisée par l'application. Pour plus d'informations, voir Activation de la sécurité pour les grilles de données.
2. Configurez l'application de grille de données afin de fournir des données d'identification utilisateur. Pour configurer les données d'identification utilisateur, vous devez utiliser un fichier `client.properties`. Vous pouvez utiliser le fichier `sampleClient.properties` qui se trouve dans le répertoire *rép_base_wxs*`/properties` d'une installation WebSphere eXtreme Scale Client pour créer votre fichier de propriétés. Apportez les modifications suivantes au fichier `client.properties` :
   - **securityEnabled** : Affectez à la propriété **securityEnabled** la valeur `true` (valeur par défaut) pour activer la sécurité client, ce qui inclut l'authentification.
   - **credentialAuthentication** : Affectez à la propriété **credentialAuthentication** la valeur `Supported` (valeur par défaut), qui indique que le client prend en charge l'authentification de données d'identification.
   - **transportType** : Affectez à la propriété **transportType** la valeur `TCP/IP`, ce qui signifie qu'aucune couche SSL (Secure Sockets Layer) n'est utilisée.
   - **singleSignOnEnabled** : Affectez à la propriété **singleSignOnEnabled** la valeur `false` (valeur par défaut). La connexion unique (Single sign-on) n'est pas disponible.
   - Les données d'identification d'un utilisateur autorisé à accéder à la grille de données :

     ```
     credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins
     .UserPasswordCredentialGenerator
     credentialGeneratorProps=<nom_utilisateur> <motdepasse>
     ```

     Exemple :

     ```
     credentialGeneratorClass=com.ibm.websphere.objectgrid.security.plugins.builtins
     .UserPasswordCredentialGenerator
     credentialGeneratorProps=xcadmin xcadmin
     ```

   - Pour utiliser un mot de passe codé, vous pouvez faire appel à l'utilitaire **FilePasswordEncoder**. Cet utilitaire se trouve dans le répertoire *rép_base_wxs*`/ObjectGrid/bin`.

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Configuration de TLS pour les applications de grille de données
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Java

Configuration de TLS pour les applications de grille de données

Vous pouvez configurer TLS (Transport Layer Security) en modifiant ou en remplaçant le fichier de clés et le fichier de clés certifiées et en choisissant un alias de certificat pour la configuration.

Avant de commencer

- Vous devez disposer d'une application de grille de données qui utilise WebSphere DataPower XC10 Appliance.
- Vous devez disposer des droits d'accès d'administrateur du dispositif.
- Vous devez disposer d'un fichier de clés ou d'un fichier de clés certifiées avec les mots de passe associés à ajouter à la configuration du dispositif. Pour modifier le fichier de clés certifiées existant, vous pouvez le télécharger à partir du dispositif. Vous devez mettre à jour le fichier de clés certifiées à l'aide des certificats publics des clients. Pour plus d'informations, voir Configuration de TLS (Transport Layer Security) pour WebSphere Application Server..
- Vous devez avoir accès à l'outil **keytool**. Cet outil se trouve dans le répertoire *rép_base_java*`/bin`.

Pourquoi et quand exécuter cette tâche

Le dispositif doit accréditer les clients qui se connectent à la grille de données. Les paramètres TLS s'appliquent à l'interface utilisateur et aux grilles de données. Les paramètres sont appliqués à tous les dispositifs de la collectivité.

Procédure

1. Téléchargez le fichier de clés certifiées actif. Dans l'interface utilisateur, cliquez sur **Collectivité** > **Paramètres** > **TLS (Transport Layer Security)**. Cliquez sur **Télécharger un fichier de clés certifiées actif** et rappelez-vous l'emplacement dans lequel vous avez enregistré le fichier sur disque, par exemple dans le répertoire `/downloads/trustStore.jks`.
2. Si nécessaire, créez un certificat et exportez le certificat public.
   a. Créez une clé privée dans le fichier de clés. La commande ci-dessous crée le fichier de clés `key.jks` qui contient la clé "ogsample". Ce fichier de clés `key.jks` est utilisé en tant que fichier de clés certifiées SSL. Exécutez la commande suivante :

      ```
       keytool -genkey -alias ogsample -keystore key.jks -storetype JKS -keyalg rsa
      -dname "CN=ogsample, U=Your Organizational Unit, O=Your Organization, L=Your
      City, S=Your State, C=Your Country" storepass ogpass -keypass ogpass -validity
      3650
      ```

   b. Exportez le certificat public. La commande ci-dessous extrait le certificat public de la clé "ogsample" et stocke la clé dans le fichier `temp.key`.

      ```
      keytool -export -alias ogsample -keystore key.jks -file temp.key -storepass
      ogpass
      ```

3. Ajoutez le certificat client au fichier de clés certifiées. Exécutez l'outil **keytool** pour importer le certificat public client vers le fichier de clés certifiées.

   ```
   keytool -import -noprompt -alias "ogsample" -keystore /downloads/trustStore.jks
   -file temp.key -storepass xc10pass -storetype jks
   ```

4. Téléchargez les informations de fichier de clés certifiées vers le dispositif. Dans l'interface utilisateur, cliquez sur **Dispositif** > **Paramètres** > **TLS (Transport Layer Security)**. Téléchargez le fichier `/downloads/trustStore.jks` mis à jour. Cliquez sur **Soumettre les paramètres TLS** pour enregistrer votre configuration.
5. Mettez à jour le fichier `client.properties`. Pour plus d'informations sur l'emplacement de ce fichier et sur les propriétés qu'il contient, voir Fichier de propriétés du client. Définissez les propriétés ci-dessous dans le fichier `client.properties`.

   ```
   securityEnabled=true
   transportType=SSL-Required
   alias=ogsample
   contextProvider=IBMJSSE2
   protocol=TLS
   keyStoreType=JKS
   keyStore=key.jks
   ```

```
keyStorePassword=ogpass
trustStoreType=JKS
trustStore=/downloads/trustStore.jks
trustStorePassword=xc10pass
```

Que faire ensuite

- Redémarrez l'application avec la nouvelle configuration pour vous connecter à l'aide de TLS (Transport Layer Security).

Rubrique parent : Sécurité

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de la sécurité de l'interface utilisateur IBM WebSphere DataPower XC10 Appliance
Gestion des utilisateurs et des groupes
Activation de la sécurité pour les grilles de données
Java Configuration de TLS (Transport Layer Security) pour WebSphere Application Server.
Java Configuration d'une application de grille de données pour l'utilisation de l'authentification client
Configuration du dispositif pour authentifier les utilisateurs à l'aide d'un annuaire LDAP

Référence associée:
Passerelle REST : Configuration de la sécurité

Traitement des incidents

Pour diagnostiquer et résoudre les problèmes que vous rencontrez dans votre environnement, vous pouvez vous aider des journaux et des traces, des données de contrôle, des données concernant le matériel et des notes sur l'édition.

Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance
Pour isoler et résoudre les problèmes rencontrés avec vos produits IBM, vous pouvez utiliser les informations relatives à l'identification et la résolution des incidents. Ces informations contiennent des consignes concernant l'utilisation des ressources de détermination des problèmes fournies avec vos produits IBM, y compris WebSphere DataPower XC10 Appliance.

Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance
Les fichiers journaux associés à WebSphere DataPower XC10 Appliance sont stockés sur le dispositif. Les journaux consultables peuvent être consultés directement à partir du dispositif avec l'interface utilisateur ou téléchargés sur votre système de fichiers local pour examen.

Téléchargement de données d'audit
L'activité d'audit est consignée par le système WebSphere DataPower XC10 Appliance. L'activité des utilisateurs concernant un ensemble d'objets sujets à audit est mémorisée de sorte à garantir une couverture adéquate de l'audit.

Analyse des données de journal et de trace
Vous pouvez utiliser les outils d'analyse de journal pour analyser le fonctionnement de l'environnement d'exécution et résoudre les problèmes qui y apparaissent.

Surveillance de la température du matériel
Des capteurs mesurent en permanence la température de divers composants internes du dispositif. Ces températures peuvent être surveillées facilement à partir d'un seul panneau à l'aide de l'interface utilisateur.

Surveillance du statut des interfaces Ethernet à l'aide d'une connexion série
Si votre système IBM® WebSphere DataPower XC10 Appliance rencontre des incidents au niveau des réseaux, vous pouvez afficher des informations sur l'activité et le statut des interfaces Ethernet. Cette rubrique décrit comment afficher ces informations à l'aide d'une connexion série établie avec le dispositif. Notez que vous pouvez aussi afficher ces informations au moyen de l'interface utilisateur.

Vérification des connexions sortantes du dispositif
A l'aide de la fonction Connexion sortante, vous pouvez vérifier si une adresse réseau est accessible à partir du dispositif.

Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif
Vous pouvez exécuter des commandes pour redémarrer le matériel du dispositif, rétablir les paramètres usine du dispositif ou arrêter le dispositif.

Traitement des problèmes d'interblocage
Cette rubrique décrit des scénarios d'interblocage courants et des solutions pour les éviter.

2.5+ **Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition**
Le scénario décrit est un exemple de transaction multipartition qui est à l'origine d'une exception de dépassement de délai de verrouillage. Selon l'état de la transaction, les solutions montrent la façon dont vous pouvez manuellement résoudre ce problème.

Traitement des problèmes d'intégration du cache
Utilisez ces informations pour traiter les problèmes de la configuration de l'intégration du cache, y compris ceux associés aux configurations de session HTTP et de cache dynamique.

Traitement des problèmes d'installation
Utilisez ces informations pour traiter les problèmes liés à l'installation et aux mises à jour.

Traitement des problèmes d'administration
Utilisez les informations suivantes pour traiter les problèmes d'administration, notamment le démarrage et l'arrêt des serveurs, en utilisant l'utilitaire **xscmd**, etc.

Notes sur l'édition
Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Pour isoler et résoudre les problèmes rencontrés avec vos produits IBM, vous pouvez utiliser les informations relatives à l'identification et la résolution des incidents. Ces informations contiennent des consignes concernant l'utilisation des ressources de détermination des problèmes fournies avec vos produits IBM, y compris WebSphere DataPower XC10 Appliance.

Techniques d'identification et de résolution d'incidents

L'*identification et la résolution des incidents* est une approche systématique de la phase de résolution d'un problème. L'objectif est de déterminer les raisons pour lesquelles une opération ne fonctionne pas comme prévu et d'expliquer la procédure à suivre pour résoudre le problème. Certaines techniques courantes peuvent être utiles pour la tâche d'identification d'incident.

Recherche dans les bases de connaissances

Vous pouvez souvent trouver des solutions à vos problèmes en effectuant des recherches dans les bases de connaissances IBM. Vous pouvez optimiser vos résultats en utilisant les ressources disponibles, les outils de support et les méthodes de recherche.

Obtention de correctifs

Il existe peut-être un correctif produit permettant de résoudre votre problème.

Comment prendre contact avec le service de support IBM

Le service de support IBM assure l'assistance relative aux problèmes rencontrés avec les produits, répond aux questions (FAQ) et aide les utilisateurs à résoudre les incidents liés au produit.

Collecte d'informations de diagnostic pour le support IBM

Vous pouvez utiliser l'interface de ligne de commande pour collecter des journaux à envoyer au support IBM.

Echange d'informations avec IBM

Pour diagnostiquer ou identifier un incident, vous avez peut-être besoin de fournir au service de support IBM des données et des informations issues de votre système. Dans d'autres cas, le service de support IBM peut vous fournir des outils ou utilitaires à utiliser pour l'identification de l'incident.

Abonnement aux mises à jour de support

Pour rester informé des informations importantes sur les produits IBM que vous utilisez, vous pouvez vous abonner aux mises à jour.

Rubrique parent : Traitement des incidents

Techniques d'identification et de résolution d'incidents

L'*identification et la résolution des incidents* est une approche systématique de la phase de résolution d'un problème. L'objectif est de déterminer les raisons pour lesquelles une opération ne fonctionne pas comme prévu et d'expliquer la procédure à suivre pour résoudre le problème. Certaines techniques courantes peuvent être utiles pour la tâche d'identification d'incident.

La première étape du processus consiste à décrire complètement le problème. La description de l'incident vous permet, et permet également au technicien de maintenance IBM, de savoir où commencer à chercher la cause de l'incident. A ce stade, vous devez vous poser les questions de base suivantes :

- Quels sont les symptômes du problème ?
- Où survient le problème ?
- Quand survient le problème ?
- Dans quelles conditions le problème survient-il ?
- Le problème peut-il être reproduit ?

Les réponses à ces questions permettent généralement de décrire avec précision le problème, ce qui vous permet de le résoudre.

Quels sont les symptômes du problème ?

Lorsque vous commencez à décrire un problème, la question la plus évidente est "Quel est le problème ?" Cette question peut sembler simple ; toutefois, vous pouvez la scinder en plusieurs questions plus concentrées qui permettent d'avoir une vue plus descriptive du problème. Ces questions sont notamment les suivantes :

- Qui ou qu'est-ce qui a généré un rapport de problème ?
- Quels sont les codes et les messages d'erreur ?
- Comment le système a échoué ? Par exemple, boucle, arrêt, plantage, dégradation des performances, résultat incorrect.

Où survient le problème ?

Il n'est pas toujours aisé de déterminer où se trouve le problème, mais il s'agit pourtant de l'une des étapes les plus importantes. De nombreuses couches technologiques peuvent exister entre le composant qui signale l'incident et le composant défaillant. Les réseaux, la grille de données et les serveurs ne sont que quelques exemples de composants à prendre en compte lorsque vous cherchez à en savoir plus sur les problèmes rencontrés.

Les questions suivantes vous aident à vous concentrer sur l'endroit où survient le problème pour isoler la couche de problème :

- Le problème est-il spécifique à une plateforme ou à un système d'exploitation, ou concerne-t-il plusieurs plateformes ou systèmes d'exploitation ?
- L'environnement et la configuration actuels sont-ils pris en charge ?
- Tous les utilisateurs rencontrent-ils ce problème ?
- (Pour les installations multisite.) Tous les sites rencontrent-ils ce problème ?

Ce n'est pas parce qu'une couche signale le problème que celui-ci provient obligatoirement de cette couche. Pour identifier l'endroit d'où provient un problème, vous devez comprendre l'environnement dans lequel ce problème se produit. Prenez quelques instants pour décrire complètement l'environnement du problème, notamment le système d'exploitation et sa version, tous les logiciels correspondants et leur version, ainsi que les informations sur le matériel. Vérifiez que la configuration de votre environnement est prise en charge ; de nombreux problèmes peuvent être provoqués par l'utilisation de logiciels incompatibles qui ne sont pas destinés à être exécutés ensemble ou dont l'exécution simultanée n'a pas été testée.

Quand survient le problème ?

Dressez la liste chronologique des événements qui ont conduit à l'apparition de l'incident, en particulier si l'incident ne s'est produit qu'une seule fois. Vous pouvez très aisément dresser une liste chronologique en procédant à l'envers : partez du moment où une erreur a été signalée (aussi précisément que possible, même à la milliseconde près), et remontez en arrière à l'aide des journaux et des informations disponibles. Il vous suffit généralement de remonter jusqu'au premier événement suspicieux signalé dans un journal de diagnostic.

Pour dresser une liste chronologique détaillée des événements, répondez aux questions suivantes :

- Le problème se produit-il uniquement à certains moments du jour ou de la nuit ?
- Selon quelle fréquence le problème se produit-il ?
- Quelle séquence d'événements a provoqué la survenue du problème ?

- Le problème s'est-il produit après un changement apporté à l'environnement, tel qu'une mise à niveau ou l'installation d'un logiciel ou d'un matériel ?

En répondant à ces questions, vous définissez un cadre de référence dans lequel mener vos recherches.

Dans quelles conditions le problème survient-il ?

Il est très important de savoir quelles applications et quels systèmes étaient en cours d'exécution lorsque l'incident s'est produit. Les questions suivantes concernant l'environnement vous aideront à identifier la cause de l'incident :

- Le problème se produit-il toujours lorsque vous effectuez la même tâche ?
- Est-ce qu'une séquence d'événements doit se produire pour provoquer le problème ?
- Est-ce que l'exécution d'autres applications échoue également ?

En répondant à ces questions, vous pouvez décrire l'environnement dans lequel l'incident se produit, et identifier les éventuelles dépendances. Notez cependant que si plusieurs incidents se produisent de manière quasi-simultanée, cela ne signifie pas nécessairement que ces incidents sont liés.

Le problème peut-il être reproduit ?

Du point de vue de l'identification et de la résolution, le problème idéal est celui qui peut être reproduit. En général, lorsqu'un problème peut être reproduit, vous disposez d'un plus grand nombre d'outils ou de procédures pour en savoir plus sur ces problèmes. Par conséquent, les problèmes que vous pouvez reproduire sont souvent plus faciles à déboguer et résoudre.

Toutefois, ces problèmes présentent un inconvénient : si le problème en question a un impact commercial considérable, vous ne voulez pas qu'il se reproduise. Si possible, recréez l'incident dans un environnement de test ou de développement qui vous offre généralement plus de souplesse et de contrôle.

- Le problème peut-il être recréé sur un système de test ?
- Est-ce que plusieurs utilisateurs ou applications rencontrent le même type de problème ?
- Le problème peut-il être recréé en exécutant une seule commande, un jeu de commandes ou une application particulière ?

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Recherche dans les bases de connaissances

Vous pouvez souvent trouver des solutions à vos problèmes en effectuant des recherches dans les bases de connaissances IBM. Vous pouvez optimiser vos résultats en utilisant les ressources disponibles, les outils de support et les méthodes de recherche.

Pourquoi et quand exécuter cette tâche

Vous pouvez trouver des informations utiles en effectuant des recherches dans le centre de documentation WebSphere DataPower XC10 Appliance. Cependant, il est parfois nécessaire de pousser les recherches plus loin pour trouver des réponses à vos questions ou résoudre certains problèmes.

Procédure

Pour rechercher les informations dont vous avez besoin dans les bases de connaissances, utilisez une ou plusieurs des approches suivantes :

- Recherchez un contenu à l'aide d'IBM® Support Assistant (ISA).

  ISA est un plan de travail de serviçabilité gratuit qui vous permet de répondre à certaines questions et de résoudre les problèmes relatifs aux logiciels IBM. Vous trouverez les instructions de téléchargement et d'installation d'ISA sur le site Web d'ISA.

- Recherchez le contenu dont vous avez besoin à l'aide du portail de support IBM.

  Le portail de support IBM est une vue centralisée et unifiée de tous les outils de support technique et de toutes les informations relatives à l'ensemble des systèmes, logiciels et services IBM. Le portail de support IBM permet d'accéder au portefeuille d'assistance électronique d'IBM à partir d'un emplacement unique. >Vous pouvez en personnaliser les pages pour cibler les informations et les ressources dont vous avez besoin pour empêcher un problème ou en résoudre plus rapidement un. Familiarisez-vous avec le portail de support IBM en visionnant les vidéos de démonstration (https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos) sur cet outil. Ces vidéos vous présentent le portail de support IBM, explorent la fonction d'identification et de résolution des problèmes et d'autres ressources et expliquent comment personnaliser la page en déplaçant, ajoutant et supprimant des portlets.

- Recherchez du contenu relatif à WebSphere DataPower XC10 Appliance en utilisant l'une des ressources techniques supplémentaires suivantes :
  - WebSphere DataPower XC10 Appliance - Notes sur l'édition
  - Site Web d'assistance WebSphere DataPower XC10 Appliance
  - Forum WebSphere DataPower XC10 Appliance
- Recherchez un contenu à l'aide de la fonction de recherche générique IBM. Vous pouvez utiliser la fonction de recherche générique IBM en saisissant votre chaîne de recherche dans la zone de recherche, dans la partie supérieure de toute page ibm.com.
- Recherchez un contenu à l'aide de tout moteur de recherche externe, tel que Google, Yahoo ou Bing. Si vous utilisez un moteur de recherche externe, vos résultats risquent d'inclure davantage d'informations en dehors du domaine ibm.com. Cependant, en consultant certains forums et blogues en dehors du domaine ibm.com, vous pouvez parfois y trouver des informations utiles pour résoudre des problèmes sur les produits IBM.

  Conseil : Incluez "IBM" et le nom du produit dans votre recherche si vous recherchez des informations sur un produit IBM.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Obtention de correctifs

Il existe peut-être un correctif produit permettant de résoudre votre problème.

Procédure

Pour rechercher et installer des correctifs, procédez comme suit :

1. Procurez-vous les outils requis pour obtenir le correctif. Utilisez IBM Update Installer pour installer et appliquer plusieurs types de packages de maintenance pour WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client. Ce programme d'installation de mises à jour étant soumis à des opérations de maintenance régulières, veuillez utiliser sa dernière version.
2. Déterminez le correctif dont vous avez besoin. Pour sélectionner le correctif le plus récent, voir Recommended fixes for WebSphere DataPower XC10 Appliance. Lors de la sélection d'un correctif, le document de téléchargement pour ce correctif s'ouvre.
3. Téléchargez le correctif. Dans le document de téléchargement, cliquez sur le lien relatif au correctif le plus récent dans la section "Download package".
4. Appliquez le correctif. Suivez les instructions de la section "Installation Instructions" du document de téléchargement.
5. Abonnez-vous pour recevoir des notifications hebdomadaires par courrier électronique sur les correctifs et d'autres informations du service de support IBM.

   Obtention de correctifs à partir de Fix Central
   Vous pouvez utiliser Fix Central pour rechercher les correctifs recommandés par le service de support IBM pour divers produits, y compris WebSphere DataPower XC10 Appliance. Avec Fix Central, vous pouvez rechercher, sélectionner, commander et télécharger des correctifs pour votre système et avez le choix entre plusieurs options de distribution. Un correctif de produit WebSphere DataPower XC10 Appliance peut être disponible pour résoudre votre incident.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Obtention de correctifs à partir de Fix Central

Vous pouvez utiliser Fix Central pour rechercher les correctifs recommandés par le service de support IBM pour divers produits, y compris WebSphere DataPower XC10 Appliance. Avec Fix Central, vous pouvez rechercher, sélectionner, commander et télécharger des correctifs pour votre système et avez le choix entre plusieurs options de distribution. Un correctif de produit WebSphere DataPower XC10 Appliance peut être disponible pour résoudre votre incident.

Procédure

Pour rechercher et installer des correctifs, procédez comme suit :

1. Procurez-vous les outils requis pour obtenir le correctif. Si votre programme d'installation de mises à jour n'est pas installé, procurez-le vous. Vous pouvez télécharger le programme d'installation à partir de Fix Central. Ce site fournit des instructions de téléchargement, d'installation et de configuration pour le programme d'installation de mises à jour.
2. Sélectionnez le produit, puis cochez une ou plusieurs cases correspondant à l'incident que vous souhaitez résoudre.
3. Identifiez et sélectionnez le correctif requis.
4. Téléchargez le correctif.
    a. Ouvrez le document de téléchargement, puis suivez le lien indiqué dans la section "Download Package".
    b. Lorsque vous téléchargez le fichier, vérifiez que le nom du fichier de maintenance n'est pas modifié. Cette modification peut être intentionnelle ou découler de certains navigateurs Web ou utilitaires de téléchargement.
5. Appliquez le correctif.
    a. Suivez les instructions de la section "Installation Instructions" du document de téléchargement.
    b. Pour plus d'informations, reportez-vous à la rubrique "Installing fixes with the Update Installer" dans la documentation du produit.
6. Facultatif : Abonnez-vous pour recevoir des notifications hebdomadaires par courrier électronique sur les correctifs et d'autres mises à jour du service de support IBM.

Rubrique parent : Obtention de correctifs

Comment prendre contact avec le service de support IBM

Le service de support IBM assure l'assistance relative aux problèmes rencontrés avec les produits, répond aux questions (FAQ) et aide les utilisateurs à résoudre les incidents liés au produit.

Avant de commencer

Une fois que vous avez essayé de rechercher votre réponse ou votre solution à l'aide d'autres options d'auto-assistance, telles que les notes sur l'édition, vous pouvez contacter le service de support IBM. Pour contacter le service de support IBM, votre société ou organisation doit avoir souscrit en contrat de maintenance IBM en cours de validité, et vous devez être autorisé à soumettre des problèmes à IBM. Pour obtenir des informations sur les types de support disponibles, reportez-vous à la rubrique Support portfolio du "*Software Support Handbook*".

Procédure

Pour signaler un problème au service de support IBM, procédez comme suit :

1. Définissez la nature du problème, collectez des informations générales, puis identifiez le niveau de gravité du problème. Pour plus d'informations, reportez-vous à la rubrique Getting IBM support du *Software Support Handbook*.
2. Collectez des informations de diagnostic.
3. Soumettez le problème au service de support IBM de l'une des manières suivantes :
    - Avec IBM® Support Assistant (ISA). .
    - En ligne via le Portail de support IBM : Vous pouvez ouvrir, mettre à jour et afficher toutes vos demandes de service à partir du portlet Demande de service de la page Demande de service.
    - Par téléphone : Pour connaître le numéro à appeler dans votre pays ou région, reportez-vous à la page Web Directory of worldwide contacts.

Résultats

Si le problème signalé concerne un incident logiciel, ou de la documentation manquante ou incorrecte, le service de support IBM crée un rapport officiel d'analyse de programme (APAR). Ce rapport expose le problème en détail. Dans la mesure du possible, le service de support IBM propose une solution palliative que vous pouvez mettre en oeuvre en attendant la finalisation de l'APAR et la mise à disposition d'un correctif . IBM publie chaque jour sur le site Web du service de support IBM, les APAR traités afin que les utilisateurs susceptibles de rencontrer le même problème puissent prendre connaissance de la solution à appliquer.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Collecte d'informations de diagnostic pour le support IBM

Vous pouvez utiliser l'interface de ligne de commande pour collecter des journaux à envoyer au support IBM.

Avant de commencer

Vous devez disposer d'une connexion d'interface de ligne de commande. Pour plus d'informations, voir Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif.

Procédure

- Collectez uniquement les informations d'état et de configuration du dispositif. Dans l'interface de ligne de commande, exécutez la commande suivante :

  ```
  platform collect-pd <PDFileName>
  ```

  Si vous n'indiquez pas un nom de fichier, le fichier `collect-pd.txt` est créé par défaut. Ce fichier contient la sortie des commandes de statut de dispositif et des détails concernant la configuration réseau.

- Collectez le journal et les fichiers de trace du dispositif. Dans l'interface de ligne de commande, exécutez la commande suivante :

  ```
  platform must-gather  <tarfilename> [<PDFileName>]
  ```

  La commande crée un fichier tar qui inclut le journal et les fichiers de trace du dispositif. Vous devez spécifier un nom de fichier tar pour exécuter la commande. Cette commande exécute la commande **platform collect-pd** avant de créer le fichier tar.

- Effacez les fichiers journaux. Si votre dispositif est en cours d'exécution depuis une longue période, les fichiers journaux peuvent être volumineux. Si le fichier tar généré par la commande **platform must-gather** est d'une taille supérieure à un gigaoctet, vous pouvez exécuter la commande suivante pour remettre à zéro l'ensemble des fichiers journaux :

  ```
  clear-logs
  ```

- Copiez les fichiers générés par le dispositif pour les envoyer au support IBM.

  1. Affichez la liste des fichiers que vous avez généré à l'aide de la commande suivante :

     ```
     Console> file list

     logsTest.tgz 1810748334 bytes created 2011-06-08 08:22:17-0500

     collect-pd.txt 17477 bytes created 2011-06-08 08:18:34-0500
     ```

  2. Copiez les fichiers journaux du dispositif avec la commande suivante :

     ```
     Console> file put logsTest.tgz
     scp://root@linux010.myco.com:/opt/cp/logsTest.tgz

     Password:*******
     ```

Que faire ensuite

Envoyez les fichiers au support IBM. Pour plus d'informations, voir Echange d'informations avec IBM.

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Echange d'informations avec IBM

Pour diagnostiquer ou identifier un incident, vous avez peut-être besoin de fournir au service de support IBM des données et des informations issues de votre système. Dans d'autres cas, le service de support IBM peut vous fournir des outils ou utilitaires à utiliser pour l'identification de l'incident.
Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Envoi d'informations au service de support IBM

Pour réduire le temps requis pour la résolution de votre incident, vous pouvez envoyer des informations de trace et de diagnostic au service de support IBM.

Procédure

Pour soumettre des informations de diagnostic au service de support IBM, procédez comme suit :

1. Ouvrez un enregistrement PMR.
2. Collectez les données de diagnostic dont vous avez besoin. Les données de diagnostic vous aident à réduire le temps de résolution de votre enregistrement PMR. Vous pouvez collecter les données de diagnostic manuellement ou automatiquement :
   - Collecte manuelle des données.
   - Collecte automatique des données.
3. Compressez les fichiers à l'aide du format de fichier `.zip` ou `.tar`.
4. Transférez les fichiers à IBM. Vous pouvez utiliser l'une des méthodes suivantes pour transférer les fichiers à IBM :
   - IBM® Support Assistant
   - L'outil de demande de service
   - Méthodes de téléchargement de données standard : FTP, HTTP
   - Méthodes de téléchargement de données sécurisées : FTPS, SFTP, HTTPS
   - Courrier électronique

   Si vous utilisez un produit z/OS et que vous faites appel à ServiceLink / IBMLink pour soumettre des enregistrements PMR, vous pouvez envoyer les données de diagnostic au service de support IBM dans un courrier électronique ou via FTP.

   Toutes ces méthodes d'échange de données sont expliquées sur le site Web du service de support IBM.

Réception d'informations du service de support IBM

Il arrive parfois qu'un technicien de maintenance IBM vous demande de télécharger des outils de diagnostic ou d'autres fichiers. Vous pouvez utiliser FTP pour télécharger ces fichiers.

Avant de commencer

Assurez-vous que votre technicien de maintenance IBM vous a fourni le serveur préféré à utiliser pour le téléchargement des fichiers, ainsi que les noms de répertoire et de fichier exacts auxquels accéder.

Procédure

Pour télécharger des fichiers à partir du service de support IBM, procédez comme suit :

1. Utilisez FTP pour vous connecter au site indiqué par votre technicien de maintenance IBM, puis connectez-vous en tant qu'utilisateur `anonymous`. Utilisez votre adresse électronique comme mot de passe :
2. Accédez au répertoire approprié :
   a. Accédez au répertoire `/fromibm`.

   ```
   cd fromibm
   ```

   b. Accédez au répertoire fourni par votre technicien de maintenance IBM.

   ```
   cd nomdurépertoire
   ```

3. Activez le mode binaire pour votre session.

   ```
   binary
   ```

4. Utilisez la commande **get** pour télécharger le fichier indiqué par votre technicien de maintenance IBM.

```
get nomfichier.extension
```

5. Mettez fin à votre session FTP.

```
quit
```

Abonnement aux mises à jour de support

Pour rester informé des informations importantes sur les produits IBM que vous utilisez, vous pouvez vous abonner aux mises à jour.

Pourquoi et quand exécuter cette tâche

En vous abonnant pour recevoir les mises à jour sur le produit, vous pouvez recevoir les mises à jour et informations techniques importantes sur des ressources et outils de support IBM spécifiques. Vous pouvez vous abonner aux mises à jour à l'aide de l'une des deux approches suivantes :

Abonnements aux média sociaux

Le flux RSS suivant est disponible pour le produit :

- Flux RSS pour le forum WebSphere DataPower XC10 Appliance

Pour des informations générales sur RSS, dont la procédure d'initiation et la liste des pages Web IBM compatibles RSS, visitez le site Flux RSS du service de support logicielIBM.

Mes notifications

Avec Mes notifications, vous pouvez vous abonner aux mises à jour de support de tout produit IBM. Mes notifications remplace Mon Support, qui est un outil similaire que vous avez peut-être utilisé dans le passé. Avec Mes Notifications, vous pouvez indiquer que vous souhaitez recevoir des annonces quotidiennes ou hebdomadaires par courrier électronique. Vous pouvez spécifier le type d'informations à recevoir (par exemple, des publications, des astuces et des conseils, des notifications flash sur les produits (ou alertes), des téléchargements et des pilotes). Mes notifications vous permet de personnaliser et classer les produits sur lesquels vous souhaitez être informé et les méthodes de distribution qui répondent le mieux à vos besoins.

Procédure

Pour vous abonner aux mises à jour de support :

1. Abonnez-vous au flux RSS pour le forum WebSphere DataPower XC10 Appliance.
    a. Dans la page d'abonnement, cliquez sur l'icône du flux RSS.
    b. Sélection l'option que vous souhaitez utiliser pour vous abonner au flux.
    c. Cliquez sur **S'abonner**.
2. Abonnez-vous à Mes notifications en accédant au Portail de support IBM® et cliquez sur **Mes notifications** dans le portlet **Notifications**.
3. Ouvrez une session à l'aide de votre ID IBM et de votre mot de passe, puis cliquez sur **Soumettre**.
4. Indiquez les types de mise à jour à recevoir et la manière de les recevoir.
    a. Cliquez sur l'onglet **S'abonner**.
    b. Sélectionnez la marque de logiciel ou le type de matériel approprié.
    c. Sélectionnez un ou plusieurs produits par nom, puis cliquez sur **Continuer**.
    d. Sélectionnez vos préférences de mode de réception des mises à jour (e-mail, en ligne dans un dossier désigné ou comme flux RSS ou Atom).
    e. Sélectionnez les types de mise à jour de documentation à recevoir, par exemple, les nouvelles informations sur les téléchargements de produit et les commentaires des groupes de discussion.
    f. Cliquez sur **Soumettre**.

Résultats

Tant que vous ne modifiez pas vos flux RSS et vos préférences Mes notifications, vous recevez les notifications des mises à jour que vous avez demandées. Vous pouvez modifier vos préférences si nécessaire (par exemple, si vous arrêtez d'utiliser un produit et commencez à en utiliser un autre).

Rubrique parent : Identification et résolution des incidents pour WebSphere DataPower XC10 Appliance

Informations connexes

- Flux RSS du service de support logiciel IBM
- S'abonner aux mises à jour de contenu du support Mes notifications
- Mes notifications pour le service de support logiciel IBM
- Présentation de Mes notifications pour le service de support logiciel IBM

Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance

Les fichiers journaux associés à WebSphere DataPower XC10 Appliance sont stockés sur le dispositif. Les journaux consultables peuvent être consultés directement à partir du dispositif avec l'interface utilisateur ou téléchargés sur votre système de fichiers local pour examen.

Avant de commencer

Vous devez disposer des droits d'accès d'administrateur du dispositif pour réaliser ces opérations.

Pourquoi et quand exécuter cette tâche

Les données de journal sont consignées directement sur le dispositif. Vous pouvez envoyer le fichier `trace.zip` à l'équipe d'assistance IBM®.

Procédure

1. Dans l'interface utilisateur, cliquez sur **Dispositif** > **Traitement des problèmes** et développez **Consignation au journal**. Les données des journaux sont stockées sur le dispositif. En développant la section **Consignation au journal**, vous pouvez accéder aux journaux disponibles à l'aide de l'afficheur de journal et vous pouvez les télécharger sur votre système de fichier pour les analyser de façon plus approfondie.
2. Visualisez les fichiers journaux en cours. Pour afficher les journaux, cliquez sur l'un des liens suivants :
   - **Afficher le fichier d'erreurs actuel**
   - **Afficher le fichier de trace actuel**
   - 2.5+ **Afficher le fichier de santé actuel**

   Une nouvelle fenêtre de navigateur s'ouvre pour l'afficheur de journal. Cet afficheur présente les 10 dernières lignes du journal sélectionné. Les nouvelles entrées au journal s'y ajoutent au fur et à mesure qu'elles sont consignées. Plusieurs actions permettent de contrôler le comportement de l'afficheur de journal.

   a. Cliquez sur **Pause** pour arrêter l'affichage de nouvelles entrées du journal. Cette action n'est disponible que si l'afficheur acceptait de nouvelles entrées.

   b. Cliquez sur **Redémarrer** pour permettre l'ajout de nouvelles entrées. Cette action n'est disponible que si l'afficheur n'acceptait plus de nouvelles entrées.

   c. Cliquez sur **Effacer** pour effacer toutes les données de l'afficheur de journal. Cette action est disponible que l'afficheur de journal accepte ou non de nouvelles entrées.
3. Cliquez sur **Télécharger les fichiers journaux** pour enregistrer tous les journaux disponibles sur votre système de fichiers. Si vous devez visualiser les informations concernant les événements qui se sont produits, vous devez utiliser ce lien. Une fenêtre s'affiche pour vous permettre d'ouvrir le fichier compressé ou de l'enregistrer sur votre système de fichiers. Le fichier`trace.zip` contient tous les fichiers journaux à fournir à l'équipe de support.

   Un ensemble de fichiers CSV est inclus dans le fichier `trace.zip` qui vous permet de suivre les données historiques des serveurs sur le dispositif. Dans le fichier `trace.zip`, les fichiers CSV se trouvent dans le répertoire *nom_serveur*`/logs`. Les fichiers sont intitulés `jvmstats.log`, `mapstats.log` et `ogstats.log`. Vous pouvez collecter les informations suivantes à partir des fichiers CSV : utilisation de la mémoire par la mappe ou le serveur, taux de réussite de la mappe, temps de transaction, débit.
4. Dans la section **Configurer les niveaux de trace**, vous pouvez afficher ou modifier les niveaux de trace. Vous pouvez modifier les niveaux de trace de la **console d'administration** ou de la **Grille de données**. Pour la console d'administration, vous pouvez modifier la sortie du gestionnaire de journalisation par défaut en la fixant aux niveaux de trace suivants :
   - OFF
   - SEVERE
   - WARNING
   - INFO
   - FINE

   a. Ajout d'une chaîne de trace. Cliquez sur **Ajouter un paramètre de trace** et ajoutez une chaîne de trace valide. Le niveau de trace pour une nouvelle chaîne de trace est défini par défaut à *INFO*.

   b. Suppression d'une chaîne de trace. Cliquez sur l'icône de suppression ( ) en regard d'une chaîne de trace pour supprimer cette dernière.

   c. Modification d'une chaîne de trace. Cliquez sur le niveau de trace et sélectionnez un nouveau niveau de trace. Cliquez sur **Sauvegarder** pour valider le nouveau niveau de trace pour la

chaîne spécifiée.

Que faire ensuite

Si la console Web n'est pas disponible, vous pouvez toujours récupérer le fichier `trace.zip`. Procédez comme suit à partir de la ligne de commande pour extraire les fichiers journaux à fournir au support IBM :

1. Collectez les fichiers journaux.

   ```
   Console> platform must-gather
   ```

2. Affichez la liste des fichiers journaux générés.

   ```
   Console> file list
   ```

3. Copiez les journaux du dispositif.

   ```
   Console> file put <nom-de-mon-choix>.tgz
   scp://root@remoteMachine.myco.com:/myRemoteMachinePath/<nom-de-mon-choix>.tgz
   ```

Envoi d'enregistrements de journal WebSphere DataPower XC10 Appliance à un système UNIX distant à l'aide de syslog
Vous pouvez activer la consignation à distance pour envoyer des entrées de journal sur un serveur distant. La consignation à distance peut être utile pour sauvegarder des informations à des fins d'audit.

Rubrique parent : Traitement des incidents

Tâches associées:
Surveillance à l'aide de fichiers CSV

Référence associée:
Définition des statistiques des fichiers CSV

Envoi d'enregistrements de journal WebSphere DataPower XC10 Appliance à un système UNIX distant à l'aide de syslog

Vous pouvez activer la consignation à distance pour envoyer des entrées de journal sur un serveur distant. La consignation à distance peut être utile pour sauvegarder des informations à des fins d'audit.

Avant de commencer

- Vous devez disposer des droits d'administration de dispositif.
- Consultez la documentation de votre système d'exploitation pour obtenir des instructions spécifiques concernant la configuration du serveur daemon syslog et la réception d'enregistrements syslog.
- WebSphere DataPower XC10 Appliance prend en charge le protocole syslog RFC 3164.
- Les enregistrements de journal sont envoyés à l'aide du protocole de datagramme utilisateur (UDP) et sont par conséquent de type "fire-and-forget", ce qui signifie que la livraison des messages n'est pas garantie (il s'agit là d'une limitation du protocole UDP).
- Notez les processus serveur de conteneur qui sont susceptibles de faire l'objet d'une consignation et d'un envoi sur le système distant sur WebSphere DataPower XC10 Appliance:
  - **`cs`**
  - **`xsServer00-xsServerXY`** : où **`XY`** représente le nombre de processus de conteneur sur le dispositif.
  - **`xsa.admin`**

  Lorsque la consignation à distance est activée, les processus peuvent être envoyés à un syslogd. Lorsqu'elle est désactivée, aucun de ces processus n'est envoyée à un syslogd. Si l'un des processus est arrêté ou temporairement indisponible, la journalisation se poursuit pour les processus qui sont encore actifs. Si vous soupçonnez des problèmes au niveau de syslog dans WebSphere DataPower XC10 Appliance qui doivent être résolus, vous pouvez ajouter un niveau de traçage **`SYSLOG=all`** via l'interface utilisateur du dispositif.

Pourquoi et quand exécuter cette tâche

Utilisez la consignation à distance pour l'archivage à long terme des événements consignés archivés. WebSphere DataPower XC10 Appliance conserve les enregistrements de journal dans un nombre restreint de fichiers journaux, pour une durée limitée afin d'économiser de l'espace. Configurez et activez la consignation à distance dans WebSphere DataPower XC10 Appliance si vous souhaitez archiver les enregistrements de journal pendant plus de temps ou si vous souhaitez analyser des enregistrements de journal archivés au-delà de la date d'expiration des journaux.

Procédure

1. Dans l'interface utilisateur du dispositif, cliquez sur **Dispositif** > **Traitement des problèmes** > **Consignation au journal**.
2. Dans la section **Configurer la consignation à distance**, sélectionner **Activer la consignation à distance**. Cela permet d'activer la consignation à distance aux fins d'analyse des données d'historique. Vous devez disposer d'un serveur syslogd qui écoute les événements et les capture. Définissez les paramètres suivants :
   a. **Hôte distant** - Indique le nom d'hôte ou l'adresse IP du serveur syslogd distant sur lequel vous voulez envoyer les enregistrements de journal. La valeur ne peut pas être "localhost", "locahost-v6", "127.0.0.1" ou "::1". Assurez-vous que vous pouvez envoyer une commande ping à l'hôte ou l'adresse IP du dispositif.
   b. **Port distant** - Indique le numéro de port du serveur syslogd sur lequel vous voulez envoyer les enregistrements de journal. Les valeurs valides sont comprises entre `0 et 65535`, et la valeur par défaut est 512.
   c. **Seuil** - Indique le seuil de gravité des messages à envoyer au serveur de journalisation distant. Pour envoyer les messages d'avertissement et d'erreur grave, entrez la valeur **WARNING**. Pour n'envoyer que les messages d'erreur grave, sélectionnez **SEVERE**.
   d. **Fonction syslog** - Indique la fonction de journalisation syslog qui est utilisée pour l'envoi des message de journal. Cette valeur détermine le fichier dans lequel le message de journal sera placé par le serveur démon syslog qui s'exécute sur le système distant. Par exemple, si vous définissez la fonction sur **user**, la plupart des serveurs démons syslog stockeront le message dans un fichier appelé `/var/log/user.log`. Si vous sélectionnez **mail**, le serveur démon syslog stockera le message dans le fichier `/var/log/mail.log`. La destination de vos enregistrements de journal dépend de la manière dont vous avez configuré le serveur démon syslog. Pour plus d'informations concernant la configuration de syslogd, reportez-vous aux instructions pour votre système d'exploitation.

   Pour modifier les paramètres, cliquez sur **Appliquer les modifications**. Les modifications sont appliquées à tous les processus concernés par les messages syslog.

Résultats

Les enregistrements de journal du dispositif sont envoyés au serveur de consignation à distance configuré pour l'archivage et l'analyse.

Rubrique parent : Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance

Téléchargement de données d'audit

L'activité d'audit est consignée par le système WebSphere DataPower XC10 Appliance. L'activité des utilisateurs concernant un ensemble d'objets sujets à audit est mémorisée de sorte à garantir une couverture adéquate de l'audit.

Pourquoi et quand exécuter cette tâche

Les données d'audit qui consignent l'activité des utilisateurs concernant des objets spécifiques sujets à audit sont stockées sur le dispositif. Reportez-vous au tableau suivant pour information sur les objets concernés inclus dans les données d'audit.

Tableau 1. Objets sujets à audit

Objets	Ajout/Création	Supprimer	Mise à jour	Autre
Intégration	Y	Y	N/D	
Groupe	Y	Y	Y	
session	N/D	N/D	N/D	Expiration, connexion
Utilisateur	Y	Y	Y	
Zone	Y	Y	Y	
Activation de zone	Y	Y	Y	

L'intégrité des données est d'une importance cruciale pour des procédures d'audit adéquates. Pour garantir l'intégrité des données, les données d'audit doivent être extraites du dispositif au moment où elles sont requises. Après avoir téléchargé les données d'audit, les méthodes utilisées pour garantir leur intégrité relèvent de votre responsabilité. Procédez comme suit pour extraire vos données d'audit du dispositif.

Procédure

1. Ouvrez la section **Audit** de l'interface utilisateur. Cliquez sur**Dispositif** > **Identification et résolution des incidents**. Développez la section **Audit**. Vous pouvez consulter les données d'audit disponibles pour votre dispositif en développant la section **Audit**.
2. Téléchargez les données d'audit. Vous pouvez télécharger l'intégralité des données ou une version filtrée.
   - **Télécharger toutes les données**

     Cliquez sur **Télécharger toutes les données** pour télécharger le fichier `audit.zip` . Le fichier `audit.zip` contient toutes les données actuellement disponibles sur le dispositif. Si vous souhaitez consulter les données d'audit au format CSV ou uniquement une partie des données disponibles, vous pouvez définir en conséquence la plage de données et télécharger les données filtrées.

   - **Télécharger les données filtrées**

     Ajustez la plage de dates et définissez les paramètres de fuseau horaire. La plage de dates et les paramètres de fuseau horaire sont utilisés pour générer les données filtrées. Ces paramètres ne s'appliquent qu'aux données filtrées et non pas lorsque vous téléchargez la totalité des données. La valeur de fuseau horaire par défaut est celle associée au dispositif. La plage de dates par défaut couvre les données des derniers 30 jours. En ajustant le fuseau horaire et la plage de dates, vous déterminez les données d'audit qui seront incluses.

     Cliquez sur **Télécharger les données filtrées** pour télécharger les données d'audit de la plage de dates spécifiée au format CSV. Le fichier `audit.csv` est un fichier au format CSV que vous pouvez télécharger sur le système de fichiers. Vous pouvez importer ces données formatées dans un logiciel fournisseur pour mise en forme complémentaire, puis les visualiser.

Rubrique parent : Traitement des incidents

Analyse des données de journal et de trace

Vous pouvez utiliser les outils d'analyse de journal pour analyser le fonctionnement de l'environnement d'exécution et résoudre les problèmes qui y apparaissent.

Pourquoi et quand exécuter cette tâche

Vous pouvez générer des rapports à partir des fichiers journaux et de trace existants dans l'environnement. Ces rapports graphiques peuvent être utilisés pour les objectifs suivants :

- **Analyse de l'état et des performances de l'environnement d'exécution :**
  - Cohérence de l'environnement de déploiement
  - Fréquence de consignation
  - Exécution de la topologie et topologie configurée
  - Modifications non planifiées de la topologie
  - Etat de quorum
  - Etat de la réplication des partitions
  - Statistiques de mémoire, rendement, utilisation du processeur, etc.
- **Pour traiter les problèmes dans l'environnement :**
  - Vues topologiques à des points spécifiques dans le temps
  - Statistiques de mémoire, rendement, utilisation du processeur au cours des problèmes
  - Niveaux de groupe de correctifs en cours, paramètres d'optimisation
  - Etat de quorum

Présentation de l'analyse du journal
Vous pouvez utiliser l'outil **`xsLogAnalyzer`** pour vous aider traiter les problèmes dans l'environnement.

Règles de stockage des fichiers journaux

Exécution de l'analyse du journal
Vous pouvez exécuter l'outil **`xsLogAnalyzer`** sur un ensemble de fichiers journaux et de trace à partir de n'importe quel ordinateur.

Création de scanners personnalisés pour l'analyse de journal
Vous pouvez créer des scanners personnalisés pour l'analyse de journal. Après avoir configuré le scanner, les résultats sont générés dans les rapports lorsque vous exécutez l'outil **`xsLogAnalyzer`**. Le scanner personnalisé recherche les enregistrements d'événement dans les journaux en fonction des expressions régulières que vous avez définies.

Traitement des problèmes d'analyse de journal
Utilisez les informations de dépannage pour identifier et éliminer les problèmes avec l'outil **`xsLogAnalyzer`** et ses rapports générés.

Rubrique parent : Traitement des incidents

Présentation de l'analyse du journal

Vous pouvez utiliser l'outil **xsLogAnalyzer** pour vous aider traiter les problèmes dans l'environnement.

Tous les messages de reprise en ligne

Affiche le nombre total de messages de reprise en ligne sous la forme d'un graphique dans le temps. Affiche également une liste des messages de reprise en ligne, y compris les serveurs qui ont été affectés.

Tous les messages critiques eXtreme Scale

Affiche les ID de message et les explications associées et les actions utilisateur qui peuvent vous faire gagner du temps dans la recherche des messages.

Toutes les exceptions

Affiche les cinq premières exceptions, y compris les messages et leur nombre et les serveurs affectés par l'exception.

Résumé de la topologie

Affiche le diagramme de la configuration de la topologie en fonction des fichiers journaux. Vous pouvez utiliser ce récapitulatif pour comparer avec votre configuration réelle, en identifiant les erreurs de configuration.

Cohérence de topologie : tableau de comparaison ORB (Object Request Broker)

Affiche les paramètres ORB de l'environnement. Vous pouvez utiliser ce tableau pour déterminer si les paramètres de l'environnement sont cohérents.

Vue Tableau chronologique d'événements

Affiche un diagramme chronologique des différentes actions qui ont eu lieu sur la grille de données, y compris les événements de cycle de vie, les exceptions, les messages critiques et les événements de diagnostic de premier niveau (FFDC).

Rubrique parent : Analyse des données de journal et de trace

Tâches associées:
Exécution de l'analyse du journal
Création de scanners personnalisés pour l'analyse de journal
Traitement des problèmes d'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Règles de stockage des fichiers journaux

Les tableaux ci-dessous décrivent les règles de stockage des fichiers journaux sur le dispositif WebSphere DataPower XC10 Appliance.

Tableau 1. Fichier de trace xsa.app

Fichier de trace xsa.app	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	5	true
audit.log	30	2097152	5	true
*stats.log	N/A	N/A	N/A	N/A

Tableau 2. Fichier de trace xsa.admin

Fichier de trace xsa.admin	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	2	true
audit.log	30	2097152	5	true
health.log	30	2097152	2	true
*stats.log	N/A	N/A	N/A	N/A

Tableau 3. Fichier cs.trace

Fichier cs	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	2	true
audit.log	30	2097152	5	true
health.log	30	2097152	2	true
*stats.log	N/A	N/A	N/A	N/A

ts.log				

*Tableau 4. Fichier de trace xsServer**

Fichier de trace xsServer*	Age maximum (en jours) de la dernière modification avant suppression	Taille maximum (en octets)	Nombre maximum de fichiers journaux	Compression si taille maximum atteinte
trace.log	30	20971520	10	true
error.log	30	2097152	2	true
audit.log	30	2097152	5	true
health.log	30	2097152	2	true
*stats.log	N/A	104857600	10	true

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Exécution de l'analyse du journal
Création de scanners personnalisés pour l'analyse de journal
Traitement des problèmes d'analyse de journal

Exécution de l'analyse du journal

Vous pouvez exécuter l'outil **xsLogAnalyzer** sur un ensemble de fichiers journaux et de trace à partir de n'importe quel ordinateur.

Avant de commencer

- Vous devez disposer d'une installation WebSphere eXtreme Scale Client. Pour plus d'informations, voir Installation de WebSphere eXtreme Scale Client.
- Collectez les fichiers journaux et de trace à partir du dispositif. Pour plus d'informations, voir Utilisation des fichiers journaux de WebSphere DataPower XC10 Appliance.
- Si vous voulez créer des scanners personnalisés, créez un fichier de propriétés de spécifications de scanner et un fichier de configuration avant d'exécuter l'outil. Pour plus d'informations, voir Création de scanners personnalisés pour l'analyse de journal.

Procédure

1. Exécutez l'outil **xsLogAnalyzer**.

   Le script se trouve dans les emplacements suivants :
   - Dans une installation autonome : *racine_install_wxs*/ObjectGrid/bin
   - Dans une installation intégrée à WebSphere Application Server : *racine_was*/bin

   Conseil : Si les fichiers journaux sont volumineux, utilisez les paramètres **-startTime**, **-endTime**, et **-maxRecords** lorsque vous exécutez le rapport pour limiter le nombre d'entrées de journal analysées. L'utilisation de ces paramètres lorsque vous exécutez le rapport améliore la clarté et l'exécution du rapport. Vous pouvez exécuter plusieurs rapports sur un même groupe de fichiers journaux.

   ```
   xsLogAnalyzer.sh|bat -logsRoot c:\myxslogs -outDir c:\myxslogs\out
   -startTime 11.09.27_15.10.56.089 -endTime 11.09.27_16.10.56.089 -maxRecords 100
   ```

   -logsRoot

   Spécifie le chemin absolu du répertoire des journaux à évaluer (requis).

   -outDir

   Spécifie un répertoire pour y placer la sortie du rapport. Si vous ne définissez pas une valeur, le rapport est écrit dans l'emplacement racine de l'outil **xsLogAnalyzer**.

   -startTime

   Spécifie l'heure de début de l'évaluation dans les journaux. La date est au format suivant : *année.mois.jour_heure.minute.seconde.milliseconde*

   -endTime

   Spécifie l'heure de fin de l'évaluation dans les journaux. La date est au format suivant : *année.mois.jour_heure.minute.seconde.milliseconde*

   -trace

   Spécifie une chaîne de trace, telle que `ObjectGrid*=all=enabled`.

   -maxRecords

   Spécifie le nombre maximal d'enregistrements pour générer le rapport. La valeur par défaut est `100`. Si vous définissez la valeur 50, les 50 premiers enregistrements sont générés pour la période définie.

2. Ouvrez les fichiers générés. Si vous n'avez pas défini de répertoire de sortie, les rapports sont générés dans le dossier `report_`*`date_time`*. Pour ouvrir la page principale des rapports, ouvrez le fichier `index.html`.
3. Utilisez les rapports pour analyser les données des journaux. Suivez les conseils ci-dessous pour optimiser les performances du rapport affiché :
   - Pour optimiser les performances des requêtes sur les données des journaux, utilisez des informations aussi spécifiques que possibles. Par exemple, la recherche de `server` dure plus longtemps et retourne plus de résultats que `server_host_name`.
   - Certaines vues ont un nombre limité de points de données affichés simultanément. Vous pouvez ajuster le segment de temps affiché en changeant les données en cours, telles que les heures de début et de fin, dans la vue.

Que faire ensuite

Pour plus d'informations sur le traitement de l'outil **xsLogAnalyzer** et les rapports générés, voir Traitement des problèmes d'analyse de journal.

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Création de scanners personnalisés pour l'analyse de journal
Traitement des problèmes d'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Création de scanners personnalisés pour l'analyse de journal

Vous pouvez créer des scanners personnalisés pour l'analyse de journal. Après avoir configuré le scanner, les résultats sont générés dans les rapports lorsque vous exécutez l'outil **xsLogAnalyzer**. Le scanner personnalisé recherche les enregistrements d'événement dans les journaux en fonction des expressions régulières que vous avez définies.

Procédure

1. Créez un fichier de propriétés de spécification de scanner qui définit l'expression générale à exécuter pour le scanner personnalisé.
    a. Créez et enregistrez un fichier de propriétés. Le fichier doit se trouver dans le répertoire *loganalyzer_root*/config/custom. Vous pouvez attribuer le nom de choix. Le fichier est utilisé par le nouveau scanner ; il est donc utile de nommer le scanner dans le fichier des propriétés. Par exemple, *my_new_server_scanner_spec*.properties.
    b. Incluez les propriétés suivantes dans le fichier *my_new_server_scanner_spec*.properties :

    ```
    include.regular_expression = REGULAR_EXPRESSION_TO_SCAN
    ```

    La variable *REGULAR_EXPRESSION_TO_SCAN* est une expression régulière en fonction de laquelle vous filtrez les fichiers journaux.

    Exemple : pour analyser les instances des lignes qui contiennent les chaînes "xception" et "rror", quel que soit l'ordre, affectez la valeur suivante à la propriété **include.regular_expression** :

    ```
    include.regular_expression = (xception.+rror)|(rror.+xception)
    ```

    Cette expression régulière permet d'enregistrer les événements si la chaîne "rror" se trouve avant ou après la chaîne "xception".

    Exemple :

    Pour analyser chaque ligne des journaux pour rechercher les lignes qui contiennent la chaîne "xception" ou "rror" quel que soit l'ordre, affectez la valeur suivante à la propriété **include.regular_expression** :

    ```
    include.regular_expression = (xception)|(rror)
    ```

    Cette expression régulière permet d'enregistrer les événements si la chaîne "rror" ou "xception" existe.
2. Créez un fichier de configuration que l'outil **xsLogAnalyer** utilise pour créer le scanner.
    a. Créez et enregistrez un fichier de configuration. Le fichier doit se trouver dans le répertoire *loganalyzer_root*/config/custom. Vous pouvez nommer le fichier *scanner_name*Scanner.config, où *scanner_name* est le nom unique du nouveau scanner. Par exemple, vous pouvez nommer le fichier serverScanner.config
    b. Incluez les propriétés suivantes dans le fichier *scanner_name*Scanner.config :

    ```
    scannerSpecificationFiles = LOCATION_OF_SCANNER_SPECIFICATION_FILE
    ```

    La variable *LOCATION_OF_SCANNER_SPECIFICATION_FILE* est le chemin et l'emplacement du fichier de spécification que vous avez créé au cours de l'étape précédente. Par exemple : *loganalyzer_root*/config/custom/my_new_scanner_spec.properties. Vous pouvez aussi définir plusieurs fichiers de spécification de scanner en utilisant une liste d'éléments séparés par un point-virgule :

    ```
    scannerSpecificationFiles =
    LOCATION_OF_SCANNER_SPECIFICATION_FILE1;LOCATION_OF_SCANNER_SPECIFICATION_FILE2
    ```
3. Exécutez l'outil **xsLogAnalyzer**. Pour plus d'informations, voir Exécution de l'analyse du journal.

Résultats

Après avoir exécuté l'outil **xsLogAnalyzer**, le rapport contient de nouveaux onglets pour les scanners personnalisés que vous avez configurés. Chaque onglet contient les vues suivantes :

Graphiques

Graphique qui illustre les événements enregistrés. Les événements sont affichés dans leurs ordre de découverte.

Tableaux

Représentation tabulaire des événements enregistrés.

Etats récapitulatifs

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Exécution de l'analyse du journal
Traitement des problèmes d'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Traitement des problèmes d'analyse de journal

Utilisez les informations de dépannage pour identifier et éliminer les problèmes avec l'outil **xsLogAnalyzer** et ses rapports générés.

Procédure

- **Problème :** manque de mémoire lors de l'utilisation de l'outil **xsLogAnalyzer** pour générer des rapports. Exemple d'erreur possible : `java.lang.OutOfMemoryError: GC overhead limit exceeded`.

  Solution : l'outil **xsLogAnalyzer** s'exécute dans une machine JVM (Java virtual machine). Vous pouvez configurer la machine JVM pour augmenter la taille de segment avant d'exécuter l'outil **xsLogAnalyzer** en définissant certains paramètres lorsque vous exécutez l'outil. L'augmentation de la taille du segment permet de stocker plus d'enregistrements dans la mémoire JVM. Commencez avec 2 048 M en supposant que le système d'exploitation dispose d'une mémoire principale suffisante. Dans la même instance de ligne de commande dans laquelle vous voulez exécuter l'outil **xsLogAnalyzer**, définissez la taille de segment de mémoire JVM maximale :

  ```
  java -Xmx<HEAP_SIZE>m
  ```

  La valeur *HEAP_SIZE* peut être un entier et représente le nombre de mégaoctets alloués au segment de mémoire JVM.
  Par exemple, vous pouvez exécuter `java -Xmx2048m`. Si vous continuez de recevoir des messages indiquant un manque de mémoire ou que vous ne disposez pas des ressources pour allouer 2 048 Mo ou plus, limitez le nombre d'événements stockés dans le segment de mémoire. Vous pouvez limiter le nombre d'événements dans le segment de mémoire en envoyant le paramètre **-maxRecords** dans la commande **xsLogAnalyzer**.

- **Problème :** lorsque vous ouvrez un rapport généré à partir de l'outil **xsLogAnalyzer**, le navigateur se bloque et ne charge pas la page.

  Cause : les fichiers HTML générés sont trop volumineux et le navigateur ne peut pas les charger. Ces fichiers sont volumineux, car la portée des fichiers journaux que vous analysez est trop grande.

  Solution : utilisez les paramètres **-startTime**, **-endTime**, et **-maxRecords** lorsque vous exécutez l'outil **xsLogAnalyzer** pour limiter le nombre d'entrées de journal analysées. L'utilisation de ces paramètres lorsque vous exécutez le rapport améliore la clarté et l'exécution du rapport. Vous pouvez exécuter plusieurs rapports sur un même groupe de fichiers journaux.

Rubrique parent : Analyse des données de journal et de trace

Concepts associés:
Présentation de l'analyse du journal

Tâches associées:
Exécution de l'analyse du journal
Création de scanners personnalisés pour l'analyse de journal

Référence associée:
Règles de stockage des fichiers journaux

Surveillance de la température du matériel

Des capteurs mesurent en permanence la température de divers composants internes du dispositif. Ces températures peuvent être surveillées facilement à partir d'un seul panneau à l'aide de l'interface utilisateur.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

Le système IBM® WebSphere DataPower XC10 Appliance surveille la température interne de divers composants pour s'assurer que le dispositif fonctionne à une température raisonnable.

Procédure

1. Accédez à l'option **Dispositif** > **Identification et résolution des incidents**.
2. Développez l'entrée **Températures du matériel**.
3. Examinez les températures de votre matériel

   Le relevé le plus récent de chaque capteur est affiché avec trois chiffres significatifs.
   - Si la température du composant correspond à une température de fonctionnement normale, une icône verte () est affichée.
   - Si la température du composant dépasse le niveau de sécurité, une icône d'avertissement jaune () est affichée.

Résultats

A l'issue de ces étapes, vous aurez examiné les relevés de températures disponibles pour vous assurer que votre dispositif fonctionne à une température raisonnable.

Que faire ensuite

Continuez à résoudre l'incident auquel vous êtes confronté.

Rubrique parent : Traitement des incidents

Surveillance du statut des interfaces Ethernet à l'aide d'une connexion série

Si votre système IBM® WebSphere DataPower XC10 Appliance rencontre des incidents au niveau des réseaux, vous pouvez afficher des informations sur l'activité et le statut des interfaces Ethernet. Cette rubrique décrit comment afficher ces informations à l'aide d'une connexion série établie avec le dispositif. Notez que vous pouvez aussi afficher ces informations au moyen de l'interface utilisateur.

Avant de commencer

Vous devez avoir un accès physique au dispositif et vous connecter avec le compte xcadmin.

Procédure

1. Etablissez une connexion série avec le dispositif en tant qu'utilisateur `xcadmin`.

   La connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif. Si vous utilisez un PC comme console série, vous devez utiliser un programme de communication série basé sur PC Windows ou Linux. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion avec le dispositif.

   Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de pouvoir continuer.

   Les paramètres de la console série sont *9600.8.n.1*.
2. Affichez le statut de l'interface Ethernet. Utilisez la commande suivante pour afficher le statut de l'interface Ethernet :

   ```
   netif status <interface>
   ```

   Où <*interface*> correspond au nom de l'interface Ethernet à interroger. Vous trouverez ci-après un exemple de sortie :

   ```
   Console> netif status mgt0
   mgt0    generic MTU:1500 flags:UP BROADCAST RUNNING MULTICAST
           inet addr:127.0.0.1 mask:255.255.255.128
           inet6 addr: 2001:DB8:0:0:0:0:0:0/32 mask: ffff:ffff:ffff:ffff::
           ethernet MAC: 00:1a:64:88:a0:6c autoneg:on duplex:Full port:TP
               speed:100Mbps
           statistics collisions:0 multicast:13 rx_bytes:101069093
               rx_compressed:0 rx_crc_errors:0 rx_dropped:0 rx_errors:0
               rx_fifo_errors:0 rx_frame_errors:0 rx_length_errors:0
               rx_missed_errors:0 rx_over_errors:0 rx_packets:1515625
               tx_aborted_errors:0 tx_bytes:10045272 tx_carrier_errors:0
               tx_compressed:0 tx_dropped:0 tx_errors:0 tx_fifo_errors:0
               tx_heartbeat_errors:0 tx_packets:20104 tx_window_errors:0
   ```

Résultats

A l'issue des ces étapes, vous avez consulté le détail du statut des interfaces Ethernet de votre dispositif.

Rubrique parent : Traitement des incidents

Vérification des connexions sortantes du dispositif

A l'aide de la fonction Connexion sortante, vous pouvez vérifier si une adresse réseau est accessible à partir du dispositif.

Avant de commencer

Vous devez disposer de droits d'accès d'administrateur de l'appareil pour réaliser ces étapes.

Pourquoi et quand exécuter cette tâche

La fonction de connexion sortante peut aider à isoler un problème en vue de le résoudre en confirmant qu'une adresse réseau cible est accessible à partir du dispositif. Cet outil n'est pas forcément pertinent au débogage de chaque incident mais constitue une méthode pratique pour s'assurer qu'une adresse réseau cible est disponible et que la connectivité avec cette adresse n'est pas entravée par un pare-feu ou par un problème réseau.

Procédure

1. Accédez à l'option **Dispositif** > **Identification et résolution des incidents**.
2. Développez l'entrée **Connexions sortantes**.
3. Entrez une adresse IP ou un nom d'hôte complet. L'adresse réseau saisie dans cette zone est utilisée en tant qu'adresse cible lorsque la commande ping est émise.
4. Cliquez sur **Commande Ping** pour lancer cette commande de sondage sur l'adresse réseau indiquée.
    - Si la tentative de connexion à l'adresse réseau saisie aboutit, l'icône suivante est affichée : .
    - Si la tentative de connexion à l'adresse réseau saisie n'a pas abouti, l'icône suivante est affichée : .

Résultats

A l'issue de ces étapes, vous aurez déterminé si une adresse cible est actuellement accessible sur le réseau à partir du dispositif.

Que faire ensuite

Continuez à résoudre l'incident auquel vous êtes confronté.

Rubrique parent : Traitement des incidents

Utilisation de l'interface de ligne de commande pour exécuter des opérations sur votre dispositif

Vous pouvez exécuter des commandes pour redémarrer le matériel du dispositif, rétablir les paramètres usine du dispositif ou arrêter le dispositif.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'interface de ligne de commande pour réinitialiser les données de configuration du dispositif, redémarrer le dispositif ou l'arrêter.

Procédure

1. Etablissez une connexion avec le dispositif en utilisant l'utilisateur `xcadmin`. Vous pouvez utiliser une connexion série ou vous connecter via l'interface de ligne de commande à l'aide d'un shell sécurisé (SSH).
    - **Connexion série :** La connexion série doit relier un terminal ASCII ou un PC doté d'un logiciel d'émulation de terminal au port série du dispositif. Si vous utilisez un PC comme console série, vous devez utiliser un programme de communication série basé sur PC pour Windows ou Linux. Utilisez le câble série fourni ou le câble USB/série PL-2303 pour établir la connexion avec le dispositif.

      Important : Si vous utilisez le câble USB/série PL-2303, téléchargez et installez un pilote pour le câble avant de continuer.
    - **Connexion éloignée :** Pour vous connecter au dispositif à l'aide de SSH, spécifiez l'URL de votre dispositif à votre client SSH.
2. Une fois la connexion avec le dispositif établie, vous pouvez exécuter des commandes.

   2.5+ Pour obtenir des informations de référence concernant les commandes que vous pouvez exécuter, voir Références des commandes de l'interface CLI.

Que faire ensuite

Si vous reconfigurez le dispositif, vous pouvez le réinitialiser. Pour plus d'informations sur la configuration du dispositif, reportez-vous à la rubrique Initialisation et configuration d'IBM WebSphere DataPower XC10 Appliance.

Rubrique parent : Traitement des incidents

Tâches associées:
Modification d'une interface d'agrégation
Suppression d'une interface d'agrégation
Ajout d'une interface d'agrégation
Gestion des interfaces Ethernet du système IBM WebSphere DataPower XC10 Appliance
Arrêt ou redémarrage du dispositif à partir de l'interface utilisateur
Contrôle des activités à l'aide des tâches

Traitement des problèmes d'interblocage

Cette rubrique décrit des scénarios d'interblocage courants et des solutions pour les éviter.

Avant de commencer

Mettez en oeuvre la gestion des exceptions dans votre application. Pour plus d'informations, voir Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java et Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications .NET.

L'exception suivante s'affiche :

```
com.ibm.websphere.objectgrid.plugins.LockDeadlockException: Message
```

Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

Procédure

- **Problème :** Une exception LockTimeoutException se produit.

  Description : Lorsqu'une transaction ou un client demande qu'un verrou soit octroyé pour une entrée de mappe spécifique, le système attend généralement que le client en cours libère le verrou avant d'envoyer la demande. Si la demande de verrou reste inactive pendant une longue période et qu'aucun verrou n'est jamais accordé, l'exception LockTimeoutException est créée pour empêcher un interblocage, ce qui est décrit plus en détail dans la section suivante. Il est plus probable que vous receviez cette exception lorsque vous configurez une stratégie de verrouillage pessimiste, car le verrou n'est jamais libéré avant la validation de la transaction.

  Extraire plus de détails :

  - Java L'exception LockTimeoutException contient la méthode getLockRequestQueueDetails, qui renvoie une chaîne. Vous pouvez utiliser cette méthode pour visualiser la description détaillée de la situation qui déclenche l'exception. Voici un exemple de code qui intercepte l'exception et affiche un message d'erreur :

    ```
    try {
        ...
    }
    catch (LockTimeoutException lte) {
        System.out.println(lte.getLockRequestQueueDetails());
    }
    ```

    Si vous recevez l'exception dans un bloc catch d'exception ObjectGridException, le code ci-dessous détermine l'exception et affiche les détails de la file d'attente. Il utilise également la méthode de l'utilitaire findRootCause.

    ```
    try {
    ...
    }
    catch (ObjectGridException oe) {
        Throwable Root = findRootCause( oe );
        if (Root instanceof LockTimeoutException) {
            LockTimeoutException lte = (LockTimeoutException)Root;
            System.out.println(lte.getLockRequestQueueDetails());
        }
    }
    ```

  - .NET L'exception LockTimeoutException contient la méthode getMessage, qui renvoie une chaîne. Vous pouvez utiliser cette méthode pour visualiser la description détaillée de la situation qui déclenche l'exception.

  Solution : une exception LockTimeoutException empêche les blocages possibles dans votre application. Une exception de ce type se produit lorsque l'application attend pendant un laps de temps défini. Vous pouvez définir la durée maximum pendant laquelle l'application attend en définissant un délai de verrouillage. Si aucun interblocage réel n'existe dans l'application, ajustez le délai d'attente de verrouillage pour éviter l'exception LockTimeoutException.

  Configurer le délai de verrouillage à l'aide d'un programme.

  - Java Configuration et mise en oeuvre du verrouillage dans les applications Java

    - .NET Configuration et mise en oeuvre du verrouillage dans les applications .NET

.

- **Problème :** Un interblocage se produit sur une clé unique.

  Description : Les scénarios suivants décrivent comment des interblocages peuvent se produire lors de l'accès à une clé unique avec un verrou S et que cette clé est mise à jour ultérieurement. Lorsque deux transactions effectuent simultanément cette action, un interblocage se produit.

  Tableau 1. Scénario d'interblocage sur clé unique

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1	get clé1	Verrou S octroyé aux deux transactions pour clé1
3	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1		Aucun verrou U. Mise à jour effectuée dans le cache transactionnel.
4		L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	Aucun verrou U. Mise à jour effectuée dans le cache transactionnel
5	Validation de la transaction		Bloquée : le verrou S pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.
6		Validation de la transaction	Interblocage : le verrou S pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 1 détient un verrou S.

  Tableau 2. Interblocages sur clé unique (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1		Verrou S octroyé pour clé1
3	L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	get clé1	Verrou S mis à niveau vers un verrou U pour clé1.
4		get clé1	Verrou S octroyé pour clé1
5		L'une des deux opérations suivantes : ◦ Java getForUpdate	Bloquée : T1 détient déjà un verrou U.

		• clé1 • .NET GetAndLock clé1	
6	Validation de la transaction		Interblocage : le verrou U pour clé1 ne peut pas être mis à niveau.
7		Validation de la transaction	Interblocage : le verrou S pour clé1 ne peut pas être mis à niveau.

Tableau 3. Interblocages sur clé unique (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1		Verrou S octroyé pour clé1
3	L'une des deux opérations suivantes : • Java getForUpdate clé1 • .NET GetAndLock clé1		Verrou S mis à niveau vers un verrou U pour clé1.
4		get clé1	Verrou S octroyé pour clé1
5		L'une des deux opérations suivantes : • Java getForUpdate clé1 • .NET GetAndLock clé1	Bloquée : unité d'exécution 1 détient déjà un verrou U.
6	Validation de la transaction		Interblocage : le verrou U pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.

Si ObjectMap.getForUpdate est utilisée pour éviter le verrou S, l'interblocage est évité :

Tableau 4. Interblocages sur clé unique (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	L'une des deux opérations suivantes : • Java getForUpdate clé1 • .NET GetAndLock clé1		Verrou U octroyé à l'unité d'exécution 1 pour clé1.
3		L'une des deux opérations suivantes :	Demande de verrou U bloquée.

		◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	
4	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	<bloquée>	
5	Validation de la transaction	<bloquée>	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.
6		<libérée>	Le verrou U est finalement octroyé à l'unité d'exécution 2 pour clé1.
7		L'une des deux opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	Verrou U octroyé à l'unité d'exécution 2 pour clé2.
8		Validation de la transaction	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.

Solutions :

- Utilisez la méthode getForUpdate ou GetAndLock au lieu de get pour obtenir un verrou U au lieu d'un verrou S.
- Utilisez le niveau d'isolement de transaction lecture validée pour éviter de maintenir des verrous S. La réduction du niveau d'isolement de transaction augmente le risque de lectures non reproductibles. Toutefois, les lectures non reproductibles à partir d'un client ne sont possibles que si le cache transactionnel est explicitement invalidé par le même client.
- Java Utilisez la stratégie de verrouillage optimiste. L'utilisation de cette stratégie requiert le traitement des exceptions de conflits optimistes. (Applications Java uniquement)

- **Problème :** Un interblocage se produit sur plusieurs clés ordonnées.

  Description : Ce scénario décrit ce qui se produit si deux transactions tentent de mettre à jour directement la même entrée et maintiennent des verrous S sur d'autres entrées.

  Tableau 5. Scénario d'interblocage sur plusieurs clés

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	get clé1	get clé1	Verrou S octroyé aux deux transactions pour clé1
3	get clé2	get clé2	Verrou S octroyé aux deux transactions pour clé2
4	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1		Aucun verrou U. Mise à jour effectuée dans le cache transactionnel.
5		L'une des deux	Aucun verrou U. Mise à jour effectuée dans le cache transactionnel.

		opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	
6.	Validation de la transaction		Bloquée : le verrou S pour clé1 ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.
7		Validation de la transaction	Bloquée : le verrou S pour clé2 ne peut pas être mis à niveau car l'unité d'exécution 1 détient un verrou S.

La méthode ObjectMap.getForUpdate permet d'éviter le verrou S, et donc l'interblocage :

Tableau 6. Scénario d'interblocage sur clés multiples (suite)

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1		Verrou U octroyé à la transaction T1 pour clé1.
3		L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	Demande de verrou U bloquée.
4	get clé2	<bloquée>	Verrou S octroyé à la transaction T1 pour clé2.
5	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	<bloquée>	
6	Validation de la transaction	<bloquée>	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.
7		<libérée>	Le verrou U est finalement octroyé à la transaction T2 pour clé1.
8		get clé2	Verrou S octroyé à la transaction T2 pour clé2
9		L'une des deux opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	Verrou U octroyé à la transaction T2 pour clé2
10		Validation de la transaction	Le verrou U pour clé1 peut être mis à niveau vers un verrou X.

Solutions :

- Utilisez la méthode getForUpdate ou GetAndLock au lieu de la méthode get pour acquérir directement un verrou U pour la première clé. Cette stratégie ne fonctionne que si l'ordre des méthodes est déterministe.
- Utilisez le niveau d'isolement de transaction lecture validée pour éviter de maintenir des verrous S. Il s'agit de la solution la plus simple à mettre en oeuvre si l'ordre des méthodes n'est pas déterministe. La réduction du niveau d'isolement de transaction augmente le risque de lectures non reproductibles. Toutefois, les lectures non reproductibles ne sont possibles que si le cache transactionnel est explicitement invalidé.
- Java Utilisez la stratégie de verrouillage optimiste. L'utilisation de cette stratégie requiert le traitement des exceptions de conflits optimistes. (Applications Java uniquement.)

- **Problème :** Un interblocage se produit à cause d'un verrou U mal ordonné.

  Description : Si l'ordre dans lequel les clés sont demandées ne peut pas être garanti, un interblocage peut toujours se produire.

Tableau 7. Scénario d'erreur d'ordre pour le verrou U

	Unité d'exécution 1	**Unité d'exécution 2**	
1	Démarrage d'une transaction	Démarrage d'une transaction	Chaque unité d'exécution effectue une transaction indépendante.
2	L'une des deux opérations suivantes : ◦ Java getForUpdate clé1 ◦ .NET GetAndLock clé1	L'une des deux opérations suivantes : ◦ Java getForUpdate clé2 ◦ .NET GetAndLock clé2	Verrou U octroyé pour clé1 et clé2
3	get clé2	get clé1	Verrou S octroyé pour clé1 et clé2
4	L'une des deux opérations suivantes : ◦ Java update clé1 ◦ .NET put clé1	L'une des deux opérations suivantes : ◦ Java update clé2 ◦ .NET put clé2	
5	Validation de la transaction		Le verrou U ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 2 détient un verrou S.
6		Validation de la transaction	Le verrou U ne peut pas être mis à niveau vers un verrou X car l'unité d'exécution 1 détient un verrou S.

Solutions :

- Effectuez tout le travail avec un verrou U simple global (mutex). Cette méthode réduit les possibilités d'accès simultané mais gère tous les scénarios lorsque l'accès et l'ordre ne sont pas déterministes.
- Utilisez le niveau d'isolement de transaction lecture validée pour éviter de maintenir des verrous S. Il s'agit de la solution la plus simple à mettre en oeuvre et qui offre le plus de possibilités d'accès simultané lorsque l'ordre des méthodes n'est pas déterministe. La réduction du niveau d'isolement de transaction augmente le risque de lectures non reproductibles. Toutefois, les lectures non reproductibles ne sont possibles que si le cache transactionnel est explicitement invalidé.
- Java Utilisez la stratégie de verrouillage optimiste. L'utilisation de cette stratégie requiert le traitement des exceptions de conflits optimistes. (Applications Java uniquement.)

Rubrique parent : Traitement des incidents

Java

Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition

2.5+ Le scénario décrit est un exemple de transaction multipartition qui est à l'origine d'une exception de dépassement de délai de verrouillage. Selon l'état de la transaction, les solutions montrent la façon dont vous pouvez manuellement résoudre ce problème.

Avant de commencer

Implémentez la gestion des exceptions dans l'application. Pour plus d'informations, voir Mise en oeuvre du traitement des exceptions dans les scénarios de verrouillage pour les applications Java.

L'exception suivante s'affiche :

```
Caused by: com.ibm.websphere.objectgrid.LockTimeoutException: Local-40000139-DEF8-05EA-
E000-64A856931719 timed out waiting for lock mode S to be granted for map name: TS2_MapP,
key: key12
granted = X
lock request queue
->[WXS-40000139-DEF6-FA84-E000-1CB456931719, state = Granted, requested 73423 milli-
seconds ago, marked to keep current mode false, snapshot mode 0, mode = X, thread name =
xIOReplicationWorkerThreadPool : 29]
->[Local-40000139-DEF8-05EA-E000-64A856931719, state = Waiting for 5000 milli-seconds,
marked to keep current mode false, snapshot mode 0, mode = S, thread name =
xIOWorkerThreadPool : 28]
dump of all locks for WXS-40000139-DEF6-FA84-E000-1CB456931719
Key: key12, map: TS2_MapP
strongest currently granted mode for key is X
->[WXS-40000139-DEF6-FA84-E000-1CB456931719, state = Granted, requested 73423 milli-
seconds ago, marked to keep current mode false, snapshot mode 0, mode = X, thread name =
xIOReplicationWorkerThreadPool : 29]
dump of all locks for Local-40000139-DEF8-05EA-E000-64A856931719
```

Ce message représente la chaîne transmise en tant que paramètre lorsque l'exception est créée et émise.

Procédure

Problème : vous obtenez une exception de dépassement de délai de verrouillage et le détenteur du verrou est une transaction multipartition, ou le dossier du journal augmente avec les messages de journal.

Diagnostic :

un message apparaît de manière répétée jusqu'à remplir le dossier des journaux, par exemple :

```
00000099 TransactionLog I  CWOBJ8705I: Automatic resolution of transaction WXS-40000139-
DF01-216D-E002-1CB456931719 at RM:TestGrid:TestSet2:20 is still waiting for a decision.
Another attempt to resolve the transaction will occur in 30 seconds.
```

Identifiez le type de transaction à l'origine du verrou. Si le préfixe dans l'identificateur de transaction est WXS-, cela indique qu'il s'agit d'une transaction multipartition. Si le préfixe dans l'identificateur de transaction est Local-, cela indique qu'il s'agit d'une transaction à une partition.

Cause : il est fort probable que l'application détienne le verrou, car aucune validation ou annulation n'a eu lieu.

Solution : déterminez l'état de la transaction et la durée de l'état. Utilisez l'utilitaire de commande **`xscmd -c listindoubts`** avec l'option `-d` (pour une sortie détaillée) ou le bean géré de la transaction.

> 2.5+ **Résolution des exceptions de délai d'attente de verrouillage**
> En utilisant la commande **`xscmd -c listindoubt`**, vous pouvez afficher l'état d'une transaction et déterminer l'action à exécuter.

Rubrique parent : Traitement des incidents

Concepts associés:
Java Stratégies de verrouillage
Java Validation en deux phases et reprise sur incident

Tâches associées:

Java Résolution des exceptions de délai d'attente de verrouillage

Java

Résolution des exceptions de délai d'attente de verrouillage

2.5+ En utilisant la commande **`xscmd -c listindoubt`**, vous pouvez afficher l'état d'une transaction et déterminer l'action à exécuter.

Rubrique parent : 2.5+ Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition

Concepts associés:
Java Stratégies de verrouillage
Java Validation en deux phases et reprise sur incident

Tâches associées:
Java Traitement des incidents liés aux exceptions de dépassement de délai d'attente pour une transaction multipartition

Résolution des exceptions de délai d'attente de verrouillage à l'aide de la commande `xscmd -c listindoubts`

Procédure

- Affichez la liste détaillée des transactions dans votre environnement : `xscmd -c listindoubt -d`
- Exécutez les actions appropriées pour résoudre la transaction. **Problème :** La transaction est marquée comme validée à TM mais les RM sont en attente de validation.

```
[1] WXS-40000139-DEF8-EF60-E002-1CB456931719
Timestamp                 Partition    Role  State     Container   Resync  Attempts
-------------------------------------------------------------------------------
--
2012-09-19 10:40:19.824  TestSet1:11  TM   COMMIT    MPTBasic2_C-0   Primary  0
2012-09-19 10:40:19.824         TestSet1:7      RM      PREPARED        MPTBasic0_C-
1       Primary         0
2012-09-19 10:40:19.839         TestSet2:20     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:40:19.824  TestSet2:6   RM   PREPARED  MPTBasic0_C-1   Primary  0
```

Solution : valider les partitions de gestionnaire de ressources (RM) et ignorez la transaction.

1. Exécutez la commande suivante pour valider la partition RM dans la transaction `WXS-40000139-DEF8-EF60-E002-1CB456931719`: `xscmd -c listIndoubts -xid WXS-40000139-DEF8-EF60-E002-1CB456931719 -cm -rm`
2. Exécutez la commande suivante pour ignorer cette transaction : `xscmd -c listIndoubts -xid WXS-40000139-DEF8-EF60-E002-1CB456931719 -f`

Problème : La transaction est en attente de validation sur toutes les partitions.

```
[1] WXS-40000139-DEF6-FA84-E000-1CB456931719
Timestamp                 Partition    Role  State     Container   Resync  Attempts
-------------------------------------------------------------------------------
--
2012-09-19 10:38:11.603         TestSet1:10     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:38:11.588         TestSet1:5      TM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:38:11.603         TestSet2:11     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:38:11.619  TestSet2:13  RM   PREPARED  MPTBasic2_C-0   Primary  0
```

Solution : annulez la partition TM et les partitions RM suivantes. Ensuite ignorez la transaction.

1. Exécutez la commande suivante pour annuler la partition TM dans la transaction `WXS-40000139-DEF6-FA84-E000-1CB456931719`: `xscmd -c listIndoubts -xid WXS-40000139-DEF6-FA84-E000-1CB456931719 -r -tm`
2. Exécutez la commande suivante pour annuler les partitions RM dans cette transaction : `xscmd -c listIndoubts -xid WXS-40000139-DEF6-FA84-E000-1CB456931719 -r -rm`
3. Exécutez la commande suivante pour ignorer cette transaction : `xscmd -c listIndoubts -xid WXS-40000139-DEF6-FA84-E000-1CB456931719 -f`

Problème : La transaction est en attente de validation sur toutes les partitions RM, mais la décision

de transaction est inconnue au niveau de TM.

```
[1] WXS-40000139-DEF8-EF31-E000-1CB456931719
Timestamp                 Partition    Role  State     Container   Resync  Attempts
-------------------------------------------------------------------------------
2012-09-19 10:40:19.777         TestSet1:11     RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:40:19.792         TestSet2:5      RM      PREPARED        MPTBasic2_C-
0       Primary         0
2012-09-19 10:40:19.777 TestSet2:6   RM    PREPARED MPTBasic2_C-1 Primary  0
```

Solution : annulez les partitions RM.

- Exécutez la commande suivante pour annuler les partitions RM dans la transaction `WXS-40000139-DEF8-EF31-E000-1CB456931719` : `xscmd -c listIndoubts -xid WXS-40000139-DEF8-EF31-E000-1CB456931719 -r`

Traitement des problèmes d'intégration du cache

Utilisez ces informations pour traiter les problèmes de la configuration de l'intégration du cache, y compris ceux associés aux configurations de session HTTP et de cache dynamique.

Procédure

- **Problème :** les ID de session HTTP ne sont pas réutilisés.

  Cause : vous pouvez réutiliser les ID de session. Si vous créez une grille de données pour la persistance des sessions dans la version 7.1.1 ou une version ultérieure, la réutilisation des ID de session est automatiquement activée. Toutefois, si vous avez créé des configurations dans des versions antérieures, ce paramètre est peut être déjà défini avec une valeur incorrecte.

  Solution : vérifiez les paramètres suivants pour déterminer si vous avez activé la réutilisation des ID de session HTTP :
  - La propriété `reuseSessionId` dans le fichier `splicer.properties` doit avoir la valeur `true`.
  - La propriété personnalisée `HttpSessionIdReuse` doit avoir la valeur `true`. Cette propriété personnalisée peut être définie dans l'un des chemins suivants dans la console d'administration WebSphere Application Server :
    - **Serveurs** > ***server_name*** > **Getion de session** > **Propriétés personnalisées**
    - **Clusters dynamiques** > ***dynamic_cluster_name*** > **Modèle de serveur** > **Gestin de session** > **Propriétés personnalisés**
    - **Serveurs** > **Types de serveur** > **Serveurs d'applications WebSphere** > ***server_name***, puis sous Infrastructure du serveur, cliquez sur **Java et gestion des processus** > **Définition de processus** > **Java virtual machine** > **Propriétés personnalisées**
    - **Serveurs** > **Types de serveur** > **Serveurs d'applications WebSphere** > ***server_name*** > **Paramètres de conteneur Web** > **Conteneur Web**

  Si vous mettez à jour les valeurs des propriétés personnalisées, reconfigurez la gestion des sessions eXtreme Scale afin que le fichier `splicer.properties` détecte la modification

- **Problème :** lorsque vous utilisez un grille de données pour stocker les sessions HTTP et que la charge des transactions est élevée, le message CWOBJ0006W figure dans le fichier `SystemOut.log`.

  ```
  CWOBJ0006W: An exception occurred:
  com.ibm.websphere.objectgrid.ObjectGridRuntimeException:
  java.util.ConcurrentModificationException
  ```

  Ce message apparaît lorsque le paramètre l'application Web modifie l'objet List défini comme attribut dans la session HTTPSession.

  Solution : clonez l'attribut qui contient l'objet List modifié et placez l'attribut cloné dans l'objet session.

- **Problème :** lors de l'exécution des applications Web avec la spécification Servlet 3,0, les filtres d'application Web et les programmes d'écoute ne sont pas appelés par la gestion de session WebSphere eXtreme Scale. Par exemple, les programmes d'écoute ne sont pas rappelés lorsque les sessions sont invalidées à l'aide d'explusion conteneur distant avec WebSphere eXtreme Scale.

  Cause : WebSphere eXtreme Scale n'identifie pas les filtres ni les programmes d'écoute définis en utilisant des annotations ou un programme.

  Solution : les filtres et programmes d'écoute doivent être explicitement déclarés dans le fichier web.xml de l'application Web.

Rubrique parent : Traitement des incidents

Concepts associés:
Topologie du dispositif : collectivités, zones et grilles de données

Traitement des problèmes d'installation

Utilisez ces informations pour traiter les problèmes liés à l'installation et aux mises à jour.

Procédure

- **Problème :** lorsque vous exécutez la commande d'installation à partir d'un ordinateur distant, tel que `\\mymachine\downloads\`, le message suivant s'affiche : `CMD.EXE was started with the above path as the current directory. UNC paths are not supported. Defaulting to Windows directory.` En conséquence, l'installation ne peut pas se terminer correctement.

  Solution : mappez l'ordinateur distant à une unité réseau. Par exemple, dans Windows, vous pouvez cliquer avec le bouton droit de la souris sur **Ordinateur**, choisir **Connecter un lecteur réseau** et inclure le chemin UNC (uniform naming conventions) vers l'ordinateur distant. Ensuite, vous pouvez exécuter le script d'installation à partir du lecteur réseau avec succès. Par exemple, `y:\mymachine\downloads\WXS\install.bat`.

- **Problème :** l'installation n'aboutit pas.

  Solution : vérifiez les fichiers journaux pour savoir où l'installation a échoué. Lorsque l'installation n'aboutit pas, les fichiers journaux se trouvent dans le répertoire *racine_install_wxs*`/logs/wxs`.

- **Problème :** échec catastrophique lors de l'installation.

  Solution : vérifiez les fichiers journaux pour savoir où l'installation a échoué. Lorsque l'installation a été partiellement exécutée, le journaux se trouvent généralement dans le répertoire *user_root*`/wxs_install_logs/`.

- Windows **Problème :** si vous installez WebSphere eXtreme Scale Client sur Windows, le texte suivant peut s'afficher dans les résultats de l'installation :

  ```
  Success: The installation of the following product was successful:
  WebSphere eXtreme Scale Client.  Some configuration steps have errors.
  For more information, refer to the following log file:
  <WebSphere Application Server install root>\logs\wxs_client\install\log.txt"
  Review the installation log (log.txt) and review the deployment manager
  augmentation log.
  ```

  Solution : si vous identifiez une erreur avec le fichier `iscdeploy.sh`, vous pouvez l'ignorer. Cette erreur ne pose pas de problème.

- Linux **Problème :**

  Si vous disposez d'une installation complète et que vous tentez d'appliquer uniquement la maintenance WebSphere eXtreme Scale Client à l'aide du programme d'installation de mises à jour, le message suivant apparaît :

  ```
  Prerequisite checking has failed. Click Back to select a different package, or click
  Cancel to exit.

  Failure messages are:

  Required feature wxs.client.primary is not found.
  ```

  Si WebSphere eXtreme Scale Client est installé et que vous tentez d'appliquer un package de maintenance complet à l'aide du programme d'installation de mises à jour, le message suivant apparaît :

  ```
  Prerequisite checking has failed. Click Back to select a different package, or click
  Cancel to exit.

  Failure messages are:

  Required feature wxs.primary is not found.
  ```

  Solution : Le package de maintenance que vous installez doit correspondre au type d'installation. Téléchargez et appliquez le package de maintenance qui s'applique à votre type d'installation.

- Linux **Problème :** l'installation se bloque.

  Solution : Parfois, lors de l'installation de WebSphere eXtreme Scale sous Linux en tant qu'utilisateur

non superutilisateur, le programme d'installation peut se bloquer. Cela est probablement du au fait que le nombre maximal de fichiers ouverts est défini sur une valeur trop basse sur votre système d'exploitation Linux. Vous devez augmenter la limite autorisée dans le fichier `/etc/limits.conf` or `/etc/security/limits.conf` (l'emplacement du fichier dépend de votre distribution Linux spécifique) sur la valeur minimum 8192.

Rubrique parent : Traitement des incidents

Tâches associées:
Utilisation du programme d'installation de mises à jour pour installer des packages de maintenance

Traitement des problèmes d'administration

Utilisez les informations suivantes pour traiter les problèmes d'administration, notamment le démarrage et l'arrêt des serveurs, en utilisant l'utilitaire **xscmd**, etc.

Procédure

- **Problème :** lorsque vous exécutez une commande **xscmd**, le message suivant s'affiche :

  ```
  java.lang.IllegalStateException: Placement service MBean not available.
  []
          at
  com.ibm.websphere.samples.objectgrid.admin.OGAdmin.main(OGAdmin.java:1449)
          at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
          at
  sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
          at
  sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37
  )
          at java.lang.reflect.Method.invoke(Method.java:611)
          at com.ibm.ws.bootstrap.WSLauncher.main(WSLauncher.java:267)
  Ending at: 2011-11-10 18:13:00.000000484
  ```

  Cause :problème de connexion avec le serveur de catalogue.

  Solution : vérifiez que les serveurs de catalogue sont actifs et disponibles via le réseau. Ce message peut aussi être généré lorsque vous disposez d'un domaine de service de catalogue défini et que moins de deux serveurs de catalogue sont actifs. L'environnement n'est pas disponible tant que deux serveurs de catalogue ne sont pas démarrés.

- **Problème :** lorsque vous exécutez une commande **xscmd**, le message suivant s'affiche :

  ```
  CWXSI0066E: Unmatched argument argument_name was detected.
  ```

  Cause : Vous avez entré un format de commande non reconnu par l'utilitaire **xscmd**.

  Solution : Vérifiez le format de la commande. Ce produit peut survenir lors de l'exécution d'expressions régulières avec la commande **-c findbyKey**. Pour plus d'informations, voir Demande, affichage et invalidation des données.

Rubrique parent : Traitement des incidents

Référence

Les informations de référence sont organisées pour vous aider à rechercher rapidement des faits.

Référence à l'utilitaire xscmd
Vous pouvez utiliser la liste de commandes suivante comme référence lorsque vous utilisez l'utilitaire **xscmd**.

2.5+ **Guide de référence de l'interface de commande HTTP**

Fichier de propriétés du client
Création d'un fichier de propriétés en fonction de vos exigences pour les processus client WebSphere eXtreme Scale.

Syntaxe d'expression régulière
Une expression régulière est une chaîne structurée utilisée pour la mise en correspondances d'autres chaînes. Vous pouvez utiliser des expressions régulières pour les données d'invalidation et pour le filtrage des messages qui s'affichent dans Message Center.

Options de configuration de mappe dynamique
Vous pouvez créer des mappes supplémentaires dans une grille de données en demandant à votre application client de se connecter à la mappe spécifiquement nommée. Une fois cette connexion établie, la mappe est automatiquement créée.

Paramètres de l'interface utilisateur
Ces informations de référence décrivent les paramètres que vous pouvez afficher et configurer dans les pages de la console d'administration WebSphere Application Server et ailleurs.

Référence à l'utilitaire `xscmd`

Vous pouvez utiliser la liste de commandes suivante comme référence lorsque vous utilisez l'utilitaire **`xscmd`**.

Paramètres de commande généraux

La syntaxe générale des commandes de l'utilitaire **`xscmd`** est la suivante. Les paramètres entre crochets sont facultatifs [ ].

ATTENTION :

N'utilisez pas les commandes suivantes dans un environnement WebSphere DataPower XC10 Appliance :

- **-c releaseShard**
- **-c reserveShard**
- **-c swapShardWithPrimary**
- **-c suspendBalancing**
- **-c resumeBalancing**
- **-c teardown**
- **-c triggerPlacement**
- **-c enableForPlacement**

```
Syntaxe : xscmd -c <cmdName> | -h <cmdName> | -lc
[<cmdGroupName>] | -lcg
             [-tt <type>] [-prot <protocol>] [-trf <filePath>] [-ks
             <filePath>] [-ksp <password>] [-user <username>] [-al <alias>]
             [-cgp <property>] [-kst <type>] [-cep <endpoints>] [-ssp
             <profileName>] [-tsp <password>] [-arc <integer>] [-trs
             <traceSpec>] [-tst <type>] [-to <serverTimeout>] [-cxpv
             <provider>] [-sp <profileName>] [-pwd <password>] [-ca <support>]
             [-cgc <className>] [-ts <filePath>]

Options:
  -al,--alias <alias>                  Nom d'alias dans le fichier de clés.

  -arc,--authRetryCount <integer>        Nombre de tentatives d'authentification si
                                       les données d'identification ont expiré. Si la
                                       valeur est 0, aucune autre tentative
d'authentification
                                       n'est effectuée.

  -c,--command <cmdName>                 Définit le nom de la commande à
                                       exécuter

  -ca,--credAuth <support>             Définit le support d'authentification des
                                       données d'identification du client [Never,
                                       Supported, Required].

  -cep,--catalogEndPoints <endpoints>    Définit un ou plusieurs noeuds finaux de service
de catalogue
                                       dans le format
                                       <host>[:<listenerPort>][,<host>[:
                                       <listenerPort>]]. Noeud final par défaut :
                                       localhost:2809

  -cgc,--credGenClass <className>        Spécifie le nom de la classe qui
                                       implémente l'interface
                                       CredentialGenerator. Cette classe utilisée pour
obtenir les données d'identification des clients.

  -cgp,--credGenProps <property>         Spécifie les propriétés de la classe
                                       d'implémentation
                                       CredentialGenerator. Les propriétés sont
paramétrées
                                       sur l'objet à l'aide de la méthode
                                       setProperties(String).

  -cxpv,--contextProvider <provider>     Fournisseur de contexte. Exemples : IBMJSSE2,
                                       IBMJSSE, IBMJSSEFIPS.
```

```
  -h,--help <cmdName>                       Appelle l'aide de ligne de commande générale

  -ks,--keyStore <filePath>                 Chemin absolu du fichier de clés. Exemple :
                                           /etc/test/security/server.public

  -ksp,--keyStorePassword <password>    Spécifie le mot de passe d'accès au fichier de
clés.

  -kst,--keyStoreType <type>                Type de fichier de clés. Exemple : JKS, JCEK,
                                           PKCS12.

  -lc,--listCommands <cmdGroupName>     Liste toutes les commandes d'un groupe de
commandes

  -lcg,--listCommandGroups                  Liste tous les groupes de commandes

  -lpc,--listPrivateCommands             Liste toutes les commandes privées.

  -prot,--protocol <protocol>               Protocole. Exemples : SSL, SSLv2,
                                           SSLv3, TLS, TLSv1.

  -pwd,--password <password>                Données d'identification de sécurité de mot de
passe eXtreme
                                           Scale

  -sp,--secProfile <profileName>            Spécifie un nom de profil.

  -ssp,--saveSecProfile <profileName>   Enregistre les valeurs des paramètres de sécurité
dans un
                                           profil de sécurité.

  -to,--timeout <serverTimeout>         Délai de connexion du serveur en secondes.

  -trf,--traceFile <filePath>               Spécifie le chemin absolu du fichier de trace
                                           généré pour la sortie de la commande
                                           xscmd

  -trs,--traceSpec <traceSpec>              Spécifie la spécification de trace de la sortie
de la commande
                                           xscmd

  -ts,--trustStore <filePath>               Chemin absolu du fichier de clés certifiées.
Exemple :
                                           /etc/test/security/server.public

  -tsp,--trustStorePassword <password>   Mot de passe du fichier de clés certifiées

  -tst,--trustStoreType <type>              Type de fichier de clés certifiées. Exemples :
JKS, JCEK,
                                           PKCS12.

  -tt,--transportType <type>                Configuration de type de sécurité de couche de
                                           transport. Exemples : TCP/IP,
                                           SSL-Supported, SSL-Required.

  -user,--username <username>               Données d'identification sécurité de nom
d'utilisateur eXtreme
                                           Scale
```

Toutes les commandes

Voici la liste complète des commandes de l'utilitaire **xscmd**.

Remarque : Les noms de colonne peuvent changer (sauf pour la position et le type d'une colonne) et de nouvelles colonnes peuvent être ajoutées à la fin d'une table si une commande est améliorée de manière appropriée.

Pour obtenir davantage d'aide sur une commande ainsi que la liste de ses paramètres, entrez **xscmd -h nom_commande**. Si vous envisagez de développer un script personnalisé avec ces commandes, vous devez

rediriger le chemin d'erreur standard vers un périphérique NULL en exécutant la commande : `./xscmd.sh -c nomCommande -Options 2> /dev/null`

Exemples

Les exemples suivants montrent comment exécuter une commande à partir de l'utilitaire **xscmd** :

```
./xscmd.sh -lc * Cette commande affiche la liste de toutes les commandes disponibles

./xscmd.sh -h showMapSizes * Cette commande affiche l'aide de la commande showMapSizes
```

Important : La sortie et l'utilisation des commandes suivantes sont susceptibles d'être modifiées dans l'avenir. Si une commande est accompagnée d'un astérisque (*), cela signifie que seule la sortie de la commande est susceptible d'être modifiée :

- listHosts
- showPlacement
- placementServiceStatus
- showDomainReplicationState
- showReplicationState
- *osgiAll
- *osgiCheck
- *osgiCurrent
- *osgiUpdate
- *showLinkedPrimaries
- *triggerPlacement

```
Nom de la commande         Description
------------               -----------
balanceShardTypes      Tentative de redistribution de fragments pour que le taux de
fragments
                           primaires et réplique dans chaque
                           serveur de conteneur corresponde à un fragment.

balanceStatus          Vérifie l'état d'équilibre de la grille de données pour l'élément
                           l'ObjectGrid et le groupe de mappes indiqués.

clearGrid              Efface les données de la grille de données.

dismissLink                Se déconnecte du domaine de service de catalogue
                           spécifié.

enableForPlacement         Réactive le placement des fragments dans un conteneur désactivé
pour le placement
                           des fragments suite à l'échec
                           d'un placement de fragment.

establishLink              Se connecte au domaine de service de catalogue défini
                           avec les noeuds finaux de service de catalogue spécifiés.

findByKey              Recherche les clés correspondantes dans une mappe.

getCatTraceSpec            Extrait la spécification de trace pour tous les
                           services de catalogue connus du processus.

getNotificationFilter      Affiche les filtres de notification pour les serveurs
                           de l'environnement.

getStatsSpec           Extrait la spécification des statistiques.

getTraceSpec           Extrait la spécification de trace.

listAllJMXAddresses    Affiche toutes les adresses de serveur de bean gérés JMX.

listCoreGroups         Liste tous les groupes centraux.

listDisabledForPlacement   Liste des conteneurs qui sont désactivés pour
                           un positionnement de fragment car ce type d'opération
                           échouent.

listHosts                  Liste tous les hôtes. La syntaxe de la commande et la sortie
```

```
sont
                        susceptibles d'être modifiées ultérieurement.

listIndoubts            Liste et résout les transactions en attente de validation.

listObjectGridNames     Liste toutes les instances ObjectGrid et tous les groupes de
mappes connus.

listProfiles            Liste les profils.

listenForNotifications  S'abonne aux notifications reçues par le concentrateur
                        de messagerie. Affiche les erreurs, les avertissements
et les
                        autres messages reçus.

osgiAll                 Affiche tous les classements de services OSGi disponibles. Utilisez
l'option
                            -sn pour afficher un seul service. Seule la sortie de la
commande
                             est susceptible d'être modifiée ultérieurement.

osgiCheck                   Détermine si les classements de service OSGi spécifiés sont
                            disponibles. La sortie de la commande est
                            est susceptible d'être modifiée ultérieurement.

osgiCurrent                 Affiche tous les classements de service OSGi utilisés.
                            Utilisez l'option -sn pour afficher un seul service.
                            La sortie de la commande uniquement est
                            est susceptible d'être modifiée ultérieurement.

osgiUpdate                  Met à jour les services OSGi vers les classements spécifiés
                            La sortie de la commande uniquement est
                            est susceptible d'être modifiée ultérieurement.

overrideQuorum              Indique au serveur de catalogue de remplacer le quorum.

placementServiceStatus Affiche l'état de l'opération de placement ObjectGrid.
                           La syntaxe et la sortie de la commande sont
                           susceptibles d'être modifiées ultérieurement.

releaseShard                Libère le fragment primaire spécifié
                            du serveur de conteneur spécifié.

removeProfile               Retire le profil du système de fichiers.

reserveShard                Réserve le fragment primaire sur le
                            serveur de conteneur spécifié.

resumeBalancing             Tente d'équilibrer et autorise les tentatives futures
d'équilibrage
                            pour l'instance ObjectGrid spécifiée et l'ensemble
                            de mappes définis.

revisions                   Affiche tout l'historique de révision connu.

routetable                  Affiche la table de routage en cours.

setCatTraceSpec             Définit la spécification de trace de tous les serveurs
                            catalogue connus du processus.

setNotificationFilter       Définit le filtre de notification. Le concentrateur de
messagerie
                            traite les messages INFO, WARNING et SEVERE
                            qui correspondent à l'expression régulière.

setStatsSpec                Définit la spécification de statistiques.

setTraceSpec                Spécification de trace dans le format :
```

```
traceType1=traceLevel1=traceState1[:traceTypeN=traceLevelN=traceStateN]*

showCoreGroupMembers       Affiche tous les membres du groupe central.

showDomainReplicationState Affiche les révisions entrantes et sortantes à
                           répliquer entre les fragments primaires dans les domaines
                           liés pour chaque conteneur dans tous les domaines.
                           La syntaxe et la sortie de la commande sont susceptibles
                           d'être modifiées ultérieurement.
showInfo                   Extrait les spécifications d'environnement, y compris
                           les versions installées et les informations relatives à la JVM.

showLinkedDomains          Affiche les domaines externes liés.

showLinkedPrimaries        Affiche les fragments primaires et tous leurs
                           fragments primaires liés locaux et externes.
                           Seule la sortie de la commande est
                           susceptible d'être modifiée ultérieurement.

showMapSizes               Affiche toutes les tailles de mappe.

showNotificationHistory    Affiche les derniers erreurs, avertissements
                           messages stockés dans
                           le concentrateur de messagerie.

showPlacement              Liste tous les serveurs de conteneur et leurs fragments.
                           La syntaxe et la sortie de la commande sont
                           susceptibles d'être modifiées ultérieurement.

showProfile                Affiche le détail du profil spécifié.

showQuorumStatus           Affiche l'état du quorum du serveur de catalogue.

showReplicationState       Affiche les révisions entrantes et sortantes
                           à répliquer entre les fragments primaires et les fragments
                           pour chaque conteneur. La syntaxe de la commande et la sortie
sont
                           susceptibles d'être modifiées ultérieurement.

showTransport              Affiche le transport utilisé par le
                           domaine du service de catalogue. Les types sont notamment ORB
et eXtremeIO.

suspendBalancing           Empêche les tentatives d'équilibrage de l'ObjectGrid et du
groupe de mappes
                           définis.

swapShardWithPrimary       Remplace le fragment réplique défini du conteneur de
                           serveur spécifié
                           par son fragment primaire.

teardown                   Arrête une liste de serveurs de catalogue et de conteneur.

triggerPlacement           Déclenche une opération de placement pour l'instance
                           ObjectGrid et le groupe de mappes. La valeur
                           numInitialContainers est ignorée.
                           Seule la commande est susceptible d'être modifiée
ultérieurement.
```

Rubrique parent : Référence

Tâches associées:
Administration avec l'utilitaire xscmd
Configuration des profils de sécurité pour l'utilitaire xscmd
Demande, affichage et invalidation des données

HTTP command interface reference

With the HTTP command interface, you can run operations on your appliance, configure appliance settings, and administer data grids, collectives, and zones.

Table of contents

List of APPLIANCE commands

CreateAdminTrace	
Description:	Adds or modifies an existing administrative trace string.
Required Parameters:	**stopOnTaskFailure** **trace**

traceName	Specifies the name of the trace.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"AutoCustomLogger", "command": "CreateAdminTrace" } }
```

Command Type:

appliance

CreateAggregateInterface

De

script ion:

Creates a new Aggregate Interface.

Required Parameters:

members

Lists all ethernet interface members of this aggregation.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

primary_member

Specifies which of the aggregated ethernet interfaces is the primary one.

aggregation_policy	Specifies the aggregation policy of the interface.
address	Specifies IP address of the interface.
name	Specifies the interface name.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

{ "task": { "member": "eth0,eth1", "stopOnTaskFailure": "true", "primary_member": "eth0", "aggregation_policy": "active-backup", "address": "1:2:3:4/24", "name": "agg1", "command": "CreateAggregateInterface" } }

Command Type:

appliance

CreateDataCacheTrace

Description:

Adds or modifies an existing data cache trace string.

Required Parameters:

stopOnTask:

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Failure traceName

Specifies the name of the trace.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"autoDataCacheLogger", "command": "CreateDataCacheTrace" } }
```

Command Type

appliance

p
e:

CreateSNMPCommunity

Description:

Creates a Simple Network Management Protocol (SNMP) community.

Required Parameters:

hostRestriction

(Optional) Specifies an IP address on which to restrict SNMP communication. Communication with any other IP address or subnet is denied.

stopOnTaskFailur

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

e

c
o
m
m
u
ni
t
y
N
a
m
e

Specifies the name of the SNMP community to create.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "hostRestriction": "autorestriction", "stopOnTaskFailure":
"true", "command": "CreateSNMPCommunity", "communityName":
"autocommunity" } }
```

Command Type

appliance

p
e:

DeleteSNMPCommunity

D
e
s
cr
ip
ti
o
n:

Deletes a Simple Network Management Protocol (SNMP) community.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
ai
lu
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

c
o
m
m
u
ni
t
y
N
a
m

Specifies the name of SNMP community to delete.

e

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"DeleteSNMPCommunity", "communityName": "autocommunity" } }
```

Command Type:

appliance

EnableCreateAccount

Description:

Enables a setting that allows users to initiate the creation of their

tion: own accounts.

Required Parameters:

stopOnTaskFailure Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

enable Enables account creation. Set the value to false to disable account creation. Set the value to true to enable account creation.

Result Parameter

rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "enable": "true", "command": "EnableCreateAccount" } }
```

Command Type:

appliance

EnablePasswordReset

Description:

Enables the ability to reset the xcadmin password with a serial connection. No other credentials or SMTP messages are required.

Required Parame

Parameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
enable	Enables the password to be reset. Set the value to true to enable password resets.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

E

Example:	{ "task": { "stopOnTaskFailure": "true", "enable": "true", "command": "EnablePasswordReset" } }
Command Type:	appliance

EnableSNMP	
Description:	Enables or disables Simple Network Management Protocol (SNMP).
Required Parameters.	
stopOnTaskFa	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

failure

enable

Enables SNMP. Set the value to false to disable SNMP. Set the value to true to enable SNMP.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "enable": "true", "command":
"EnableSNMP" } }
```

Command Type:

appliance

ModifyAdminDefaultTrace

Description: Changes the trace level for the administrative default logger.

Required Parameters:

stopOnTaskFailure
Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

traceLevel
Specifies the trace level. Valid values: OFF, SEVERE, WARNING, or INFO.

Re

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceLevel": "SEVERE",
"command": "ModifyAdminDefaultTrace" } }
```

Command Type: appliance

ModifyAdminTrace
Description: Changes an administrative logger trace level.

Re

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

traceLevel

Specifies the trace level. Valid values: OFF, SEVERE, WARNING, INFO, FINE, FINER, FINEST, ALL.

traceName

Specifies the name of the trace to modify.

Result Para

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceLevel": "SEVERE",
"traceName": "autoCustomLogger", "command": "ModifyAdminTrace"
} }
```

Command Type:

appliance

ModifyAdministratorEmail

Description:

Changes the default administrator email address.

Required P

Parameters:

StopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
administratorEmail	Specifies a new email address for the default administrator account.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "administratorEmail":
"somenewname@us.ibm.com", "command":
"ModifyAdministratorEmail" } }
```

Command Type:

appliance

ModifyAdministratorName

Description:

Changes the default administrator account name.

Required Parameters:

st

opOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

administratorName

Specifies the new name for the default administrator account.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

E

Example:	{ "task": { "stopOnTaskFailure": "true", "administratorName": "somenewname", "command": "ModifyAdministratorName" } }
Command Type:	appliance

ModifyAdministratorPassword	
Description:	Changes the password for the default administrator account.
Required Parameters.	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

newPassword

Specifies a new default administrator account password.

passwordVerification

Specifies a new default administrator account password, entered a second time for verification.

Response Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "newPassword": "somepass",
"passwordVerification": "somepass", "command":
"ModifyAdministratorPassword" } }
```

Command Type:	appliance

ModifyAggregateInterface	
Description:	Modifies Aggregate Interface parameters.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
n	

name	Specifies the interface name.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:


```
{ "task": { "mtu": "1500", "member": "eth0,eth1",
"stopOnTaskFailure": "true", "transmit_hash_policy": "layer2",
"dad_retransmit_timer": "1000", "use_slaac": "true", "mode": "Auto",
"use_dhcp": "false", "use_arp": "true", "primary_member": "eth0",
"aggregation_policy": "active-backup", "address": "1:2:3:4/24",
"name": "agg1", "command": "ModifyAggregateInterface",
"dad_transmits": "1", "lacp_selection_logic": "stable",
"ipv4_default_gateway": "1:2:3:5" } }
```


Command Type:

appliance

ModifyDataCacheTrace

Des

cr
ip
ti
o
n:

Adds or modifies an existing data cache trace string.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
ai
lu
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

tr
a
c
e
L
e
v
el

Specifies the trace level. Valid values: OFF, SEVERE, WARNING, INFO, FINE, FINER, FINEST, ALL.

tr
a
c
e
N
a
m
e

Specifies the name of the trace.

R

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceLevel": "SEVERE",
"traceName": "autoDataCacheLogger", "command":
"ModifyDataCacheTrace" } }
```

Command Type:

appliance

ModifyEthernetInterface

Description:

Modifies Ethernet Interface parameters.

Result:

equiredParameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
name	Specifies the interface name.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
comma	

ndResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "mtu": "1500", "use_dhcp": "false", "use_arp": "true",
"stopOnTaskFailure": "true", "dad_retransmit_timer": "1000",
"use_slaac": "true", "address": "1:2:3:4/24", "name": "eth0",
"dad_transmits": "1", "command": "ModifyEthernetInterface",
"ipv4_default_gateway": "1:2:3:5", "mode": "Auto" } }
```

Command Type: appliance

ModifyLDAP

Description:

Configures the appliance to use Lightweight Directory Access Protocol (LDAP) for user login authentication.

Required Parameters:

jndiSecurityPri	Specifies the JNDI security principal.

nci pal	
sto pOn Tas kFai lure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
lda pUs erId Sea rch Filt erP atte rn	LDAP user search filter pattern.
ne wPa ssw ord	Specifies the new LDAP password.
gro upB ase Dn	Specifies the LDAP JNDI base distinguished name (DN) for groups.
pas swo rdV erifi cati on	Specifies the new LDAP password a second time for password verification. The value must be the same as the newPassword parameter.
use rBa seD n	Specifies the LDAP JNDI base distinguished name (DN) for users.
ena ble	Specifies if LDAP user authentication is enabled. Set the value to true to enable LDAP user authentication.
jndi Pro vid erU RL	Specifies the URL for the LDAP provider.

Result Parameter

rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

E
x
a
m
pl
e:

```
{ "task": { "stopOnTaskFailure": "true", "jndiSecurityPrincipal":
"CN=Administrator,CN=users,DC=mycompany,DC=com",
"ldapUserIdSearchFilterPattern": "thefilterpattern", "newPassword":
"somepassword", "passwordVerification": "somepassword",
"groupBaseDn": "cn=group", "userBaseDn": "cn=user", "enable":
"true", "command": "ModifyLDAP", "jndiProviderURL": "someurl" } }
```

C
o
m
m
a
n
d
T
y
p
e:

appliance

ModifyNtpServers

D
e
s
cr
ip
ti
o
n:

Changes the list of Network Time Protocol (NTP) servers.

R
e
q
ui
r
e
d
P
a
r
a
m

eters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

listOfServers

Specifies a list of comma-separated Network Time Protocol (NTP) servers.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.

commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "listOfServers":
"time.nist.gov,10.10.2.2", "command": "ModifyNtpServers" } }
```

Command Type:

appliance

ModifySMTPEmail

Description:

Changes the reply-to email address that is used for sending passwords that are reset by users.

Required Parameters:

stopOn	Specifies whether to stop running the batch routine

TaskFailure	when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
emailAddress	Specifies the reply-to email address. Use the email address for the administrator.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ModifySMTPEmail", "emailAddress": "admin@mycompany.com" } }
```

Co

Command Type:	appliance

ModifySMTPServer

Description:	Changes the Simple Mail Transfer Protocol (SMTP) server.
Required Parameters:	**stopOnTaskFailure** Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only. **host**

Name	Specifies the SMTP host address.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ModifySMTPServer", "hostName": "sicdsjc" } }
```

Command Type:

appliance

ModifySearchFilterUsers

De

Description:

Changes the Lightweight Directory Access Protocol (LDAP) search filter for users.

Required Parameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
ldapUserIdSearchFilterPattern	LDAP user search filter pattern.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true",
"ldapUserIdSearchFilterPattern": "(&(uid={0})
(objectclass=inetOrgPerson))", "command":
"ModifySearchFilterUsers" } }
```

Command Type: appliance

ModifyTimeZone

Description: Changes the time zone for the appliance.

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

localTimeZone

Specifies the new time zone as an abbreviated string. Valid values: EST : Eastern Standard Time CST : Central Standard Time MST : Mountain Standard Time PST : Pacific Standard Time HAST : Hawaii AKST : Alaska AST : Atlantic UTC : Coordinated Universal Time GMT : Greenwich Mean Time CET : Central Europe EET : Eastern Europe MSK : Moscow AST_ARABIA : Arabia KRT : Pakistan IST : India NOVT : Novosibirsk CST_CHINA : China JST : Japan AWST : Australia West ACST : Australia Central AEST : Australia East

Result Parameters:

errorMessage A message that explains the reason for the failure, if the command failed.

status Specifies the status of the command that was run. The result can be either success or failed.

commandResult A JSON-formatted statement that contains the result of the command that was run.

commandName Specifies the name of the HTTP command interface command that was run.

Example

{ "task": { "stopOnTaskFailure": "true", "command":

ple:	"ModifyTimeZone", "localTimeZone": "EST" } }
Command Type:	appliance

PingRemoteHost	
Description:	Tests the visibility of a host from this appliance.
Required Parameters:	
stopOnTaskFailur	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

e

hostName

Specifies the IP address or host name of the remote host.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"PingRemoteHost", "hostName": "io03.rtp.raleigh.ibm.com" } }
```

Command Type:

appliance

RemoveAdminTrace

Description:

Removes an existing administrative console trace string.

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

traceName

Specifies the name of the trace.

Result

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"AutoCustomLogger", "command": "RemoveAdminTrace" } }
```

Command Type: appliance

RemoveAggregateInterface

Description: Deletes Aggregate Interface of a given name.

Requi

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

name

Specifies the interface name.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "name": "agg1", "command": "RemoveAggregateInterface" } }
```

Command Type:

appliance

RemoveDataCacheTrace

Description:

Removes an existing data cache trace string.

Required Parameters:

stopOnT	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the

taskFailure	batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
traceName	Specifies the name of the trace.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "traceName":
"autoDataCacheLogger", "command": "RemoveDataCacheTrace" } }
```

Command

appliance

Type:

RestartAppliance

Description:

Restarts or shuts down the appliance.

Required Parameters:

immediate

Specifies if the appliance waits to complete active tasks before restarting. If the value is set to true, the appliance restarts immediately. If the value is set to false, the appliance waits for all active tasks to complete before restarting.

stopOnTaskFailur

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

e **s** **h** **u** **t** **d** **o** **w** **n**	Specifies if the appliance restarts or shuts down. If the value is set to true, the appliance is shut down. If the value is set to false, the appliance restarts.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "immediate": "false", "stopOnTaskFailure": "true",
"command": "RestartAppliance", "shutdown": "true" } }
```

Command Type:

appliance

ViewAllAggregateInterfaces	
Description:	Displays the information for every aggregate interface.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
Result Parameter	

e
rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

E
x
a
m
pl
e:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ViewAllAggregateInterfaces" } }
```

C
o
m
m
a
n
d
T
y
p
e:

appliance

ViewAllEthernetInterfaces

D
e
s
cr
ip
ti
o
n:

Displays the information for every Ethernet Interface.

R
e
q
ui
r
e
d
P
a
r
a

Parameters:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ViewAllEthernetInterfaces" } }
```

Command Type: appliance

List of COLLECTIVE commands

AddApplianceToCollective

Description: Adds an existing appliance to the current collective.

Required Parameters:

appliancelPos

Specifies the IP address of the appliance to add to the collective.

nTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
secretKey	Specifies the secret key of the appliance.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
zoneName	Specifies the name of the zone to which you want to assign the appliance.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "applianceIP": "192.168.222.128", "stopOnTaskFailure":
"true", "secretKey": "P5Xa4F8MQeSxuuhKxnaStw==", "waitOnTask":
"true", "command": "AddApplianceToCollective", "zoneName":
"DefaultZone" } }
```

Command Type:

collective

AddRoleToGroup

Description:

Assigns an access role to a group. The defined roles are: appliance administration, appliance monitoring, or data cache creation.

Required Parameters:

g

groupName

Specifies the name of the group that is being assigned to a role.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

role

Specifies the role to assign to the group. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Ex

Example:	{ "task": { "stopOnTaskFailure": "true", "groupName": "autoGroup1003", "command": "AddRoleToGroup", "role": "5" } }
Command Type:	collective

AddRoleToUser

Description:	Assigns an access role to a user. The defined roles are: appliance administration, appliance monitoring, or data cache creation.
Required Parameters:	
stopOnTaskFailu	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

role

Specifies the role to assign to the user. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

userName

Specifies the name of the user that is being assigned a role.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "AddRoleToUser",
"role": "2", "userName": "autoUser1003" } }
```

CommandT

collective

y
p
e:

AddUserToGroup

D
e
s
cr
ip
ti
o
n:

Adds a user to a group. After the user is added to the group, the user has the roles that are assigned to the selected group only. Any individual user roles are lost.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

g r o u p N a m e

Specifies the name of a defined group.

s t o p O n T a s k F ai lu r e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

userName	Specifies the name of a defined user.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "AddUserToGroup", "userName":
"autoUser1003" } }
```

Command Type:

collective

CreateCollectiveLink

Description:

Creates a link to another collective.

Required Parameters:

appliance Name

Specifies the host name or IP address of an appliance that is a member of another collective.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

remotePassword

Specifies the password that is used to log in to the remote appliance.

remoteUserName

Specifies the user name that is used to log in to the remote appliance.

Response Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Ex

```
{ "task": { "applianceName": "myappliance.mydomain.com",
```

Example:

"stopOnTaskFailure": "true", "command": "CreateCollectiveLink", "remotePassword": "somePassword", "remoteUserName": "someUser" } }

Command Type:

collective

CreateGroup

Description:

Creates a new group. Groups are useful for assigning multiple users the same set of roles.

Required Parameters:

groupName

st

Specifies a name for the new group. Group names must be unique.

opOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

groupDescription

Specifies a description of the group and its users.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Ex

Example:	{ "task": { "stopOnTaskFailure": "true", "groupName": "autoGroup1003", "groupDescription": "this group is huge", "command": "CreateGroup" } }
Command Type:	collective

CreateUser

Description:	Creates a new user.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
stopOnTaskFailurep	

passwordVerification	Specifies the password, entered a second time for verification.
userPassword	Specifies a new password for the selected user.
emailAddress	Specifies an email address for the new user.
fullUserName	Specifies a full name for the user. Example: John E. Doe.
userName	Specifies a short name for the new user. Example: xcadmin.

ResultParameter

rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "passwordVerification":
"someuserpass", "userPassword": "someuserpass", "command":
"CreateUser", "emailAddress": "someuser@us.ibm.com",
"fullUserName": "someuser isme", "userName": "someuser" } }
```

Command Type:

collective

DeleteCollectiveLink

Description:

Removes the link between the local collective and the specified collective.

Required Parame

Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

name

Specifies the name of the collective to delete.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example

```
{ "task": { "stopOnTaskFailure": "true", "name": "myCollectiveName",
"command": "DeleteCollectiveLink" } }
```

Command Type:	collective

DeleteGroup

Description:	Deletes a group. Users that belonged to the group no longer belong to the group and might lose any assigned roles.
Required Parameters:	
groupName	Specifies the name of a defined group.
stopOn	

TaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "DeleteGroup" } }
```

Command Type:

collective

DeleteUser

Description:

Deletes a user. If the user belongs to a group, the user is removed from the group.

Required Parameters:

StopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

UserName

Specifies the name of a defined user.

Result

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "DeleteUser",
"userName": "autoUser1003" } }
```

Command Type:

collective

GetCollectiveName

Description:

Retrieves the collective name.

Require

edParameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Ex

Example:	{ "task": { "stopOnTaskFailure": "true", "command": "GetCollectiveName" } }
Command Type:	collective

GetHealthStatus	
Description:	Displays the hardware and software Health Status information for the appliance.
Required Parameters:	
stopOnTaskFa	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

il
u
r
e

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "GetHealthStatus" } }
```

Command Type:

collective

ModifyGroupDescription

De

scription:

Changes the description of a selected group.

Required Parameters:

groupName

Specifies the name of a defined group.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

groupDesc

Specifies a description of the group and its users.

ri
p
ti
o
n

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName": "somegroup",
"groupDescription": "somegroupdesc", "command":
"ModifyGroupDescription" } }
```

Command Type:

collective

ModifyScheduleExportSettings

De

script ion:

Enable and disable the scheduled exports feature. Also, specify an interval to generate a configuration export.

Required Parameters:

daylinterval

Specifies how many days to wait in between scheduled configuration exports.

time

Specifies the scheduled date that the first export will occur. Must be in the following format: YYYY-MM-DD. For example, "2014-09-08".

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

enable

Specifies if the scheduled configuration export feature is enabled or disabled.

date	CLI.parameter.date.desc

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "dayInterval": "3", "stopOnTaskFailure": "true", "time":
"16:25", "enable": "true", "command":
"ModifyScheduleExportSettings", "date": "2014-02-14" } }
```

Command Type:

collective

ModifyTransportAndStorageMode

De

script:

Changes the transport and storage mode setting for the collective. When you update the transport and storage mode, a task is created to update the collective settings and all appliances in the collective are restarted.

Required Parameters:

transport

Specifies the transport and storage mode. Possible values are ORB, XIO, XIO_XM. Transport and storage modes enable the exchange of objects and data between different server processes. When you enable IBM eXtremeIO (XIO), relative response time is faster than the Object Request Broker (ORB). ORB is deprecated. With XIO enabled, you can also create an enterprise data grid. With an enterprise data grid, client applications that are written in different programming languages, such as Java and .NET, can access the same data grid. When you use IBM eXtremeMemory, cache entries are stored in native memory. When you use heap memory, cache entries are stored in the Java heap.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "command": "ModifyTransportAndStorageMode", "mode":
"XIO" } }
```

Command Type: collective

ModifyUserEmail

Description: Changes the email address for a user.

Required Parameters:

st

opOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

emailAddress

Specifies a new email address for the user.

userName

Specifies the user name of the user to update.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandRes

A JSON-formatted statement that contains the

ult	result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"ModifyUserEmail", "emailAddress": "user1@mycompany.com",
"userName": "user1" } }
```

Command Type: collective

ModifyUserPassword

Description: Changes the password for a specified user.

Required Parameters:

stopOn	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the

TaskFailurenewPassswordpasswordVerificationuserName	batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only. Specifies a new password for the selected user. Specifies the new password for the selected user, entered a second time for verification. Specifies the name of the user to update.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
comma	

ndResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "newPassword": "newpass",
"passwordVerification": "newpass", "command":
"ModifyUserPassword", "userName": "autoUser1003" } }
```

Command Type: collective

RemoveApplianceFromCollective

Description: Removes an existing appliance from the current collective.

Required Parameters:

appli

a
n
c
e
l
P

Specifies the IP address of the appliance to be removed.

s
t
o
p
O
n
T
a
s
k
F
a
il
u
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

w
a
it
O
n
T
a
s
k

Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.

R
e
s
ul
t
P
a
r
a
m
e
t
rs
:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
comma	

ndName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "applianceIP": "192.168.222.128", "stopOnTaskFailure":
"true", "waitOnTask": "true", "command":
"RemoveApplianceFromCollective" } }
```

Command Type:

collective

RemoveRoleFromGroup

Description:

Removes an access role from a group. The defined roles are: appliance administration, appliance monitoring, or data cache creation.

Required Parameters:

groupNa	Specifies the name of the group for which you are removing an access role.

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
role	Specifies the role to remove from the group. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "RemoveRoleFromGroup", "role": "5"
} }
```

Command Type:	collective

RemoveRoleFromUser

Description:	Removes an access role from a user. The defined roles are: appliance administration, appliance monitoring, or data cache creation.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
role	Specifies the role to remove from the user. Valid values: 2 = appliance administration, 3 = appliance monitoring, 5 = data cache creation.

username

Specifies the name of for which you are removing an access role.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command":
"RemoveRoleFromUser", "role": "2", "userName": "autoUser1003" } }
```

Command Type:

collective

RemoveUserFromGroup

Description: Removes a user from a group. After a user is removed from a group, the user continues to have the same roles as the group. However, the user roles for the user no longer change when the group roles change. You must modify the user roles for the selected user individually.

Required Parameters:

groupName Specifies the name of a defined group.

stopOnTaskFailure Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

userName Specifies the name of a defined user.

a
m
e

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "RemoveUserFromGroup",
"userName": "autoUser1003" } }
```

Command Type:

collective

ViewAllGroups

Descr

ip
ti
o
n:

Displays information about all groups.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
a
il
u
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

R
e
s
ul
t
P
a
r
a
m
e
t
e
rs
:

errorM
essage

A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewAllGroups"
} }
```

Command Type: collective

ViewAllUsers

Description: Displays the information for every user.

Required Parameters:

s

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewAllUsers" }
}
```

Command Type

collective

p
e:

ViewCollectiveLinks

D
e
s
cr
ip
ti
o
n:

Displays the information for every collective link.

R
e
q
ui
r
e
d
P
a
r
a
m
e
t
e
rs
:

s
t
o
p
O
n
T
a
s
k
F
a
il
u
r
e

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

R
e
s
ul
t
P
a
r
a

meters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewCollectiveLinks" } }
```

Command Type:

collective

ViewGroup

Description:

Displays information about a selected group.

Required Pa

Parameters:

groupName	Specifies the name of a defined group.
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

ResultParameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.

commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName":
"autoGroup1003", "command": "ViewGroup" } }
```

Command Type: collective

ViewMemberDetails

Description: Displays information about a member in a collective.

Required Parameters:

appliance	Specifies the IP Address of the member.

cIP stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "applianceIP": "192.168.222.128", "stopOnTaskFailure":
"true", "command": "ViewMemberDetails" } }
```

Co

Command Type:	collective

ViewTransportAndStorageMode	
Description:	Displays the transport and storage mode.
Required Parameters:	
Result Parameters:	

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "command": "ViewTransportAndStorageMode" } }
```

Command Type:

collective

ViewUser

Description:

Displays information about a selected user.

Required Parameters:

s

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

userName

Specifies the name of a defined user.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "command": "ViewUser",
"userName": "autoUser1003" } }
```

Command Type: collective

List of GRID commands

ClearGrid	
Description:	Clears data from a data grid.
Required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to be cleared.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "ClearGrid" } }
```

Command Type:

grid

CreateGrid

Description: Creates a new simple, dynamic cache, or session data grid.

Required Parameters:

stopOnTaskFailure — Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName — Specifies the name of the data grid to create.

waitOnTas — Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.

k

g
ri
d
T
y
p
e

Specifies the type of data grid to create. Valid values: simple, dynamic, or session.

Response Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"waitOnTask": "true", "command": "CreateGrid", "gridType": "simple"
} }
```

Command Type:

grid

DeleteGrid

Description:

Deletes a data grid.

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to delete.

waitOnTask

Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.

k

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "DeleteGrid" } }
```

Command Type:

grid

ExportGrid

Description:

(Simple data grids only) Exports data grid information to XML.

on:

Required Parameters:

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to export.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "ExportGrid" } }
```

Command Type:

grid

GrantGroupAccess

Description:

Gives a group access rights to a data grid.

Required Parameter

rs
:

groupName

Specifies the name of the group to grant the given access rights.

stopOnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to grant access rights.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

comma

A JSON-formatted statement that contains the

ndResult	result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName": "somegroup",
"gridName": "myGrid", "command": "GrantGroupAccess" } }
```

Command Type: grid

GrantUserAccess

Description: Gives a user access rights to a data grid.

Required Parameters:

stop

OnTaskFailure

Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

gridName

Specifies the name of the data grid to which the user is assigned access.

userName

Specifies the name of the user to give access rights.

Result Parameters:

errorMessage

A message that explains the reason for the failure, if the command failed.

status

Specifies the status of the command that was run. The result can be either success or failed.

commandResult

A JSON-formatted statement that contains the result of the command that was run.

commandName

Specifies the name of the HTTP command interface command that was run.

E

Example:	{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid", "command": "GrantUserAccess", "userName": "someuser" } }
Command Type:	grid

ModifyGridCapacity	
Description:	Changes the capacity for a simple data grid.
Required Parameters.	
stopOnTaskF	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.

failureUseLRUgridName	(Simple data grids only) Enables least recently used (LRU) eviction when set to true.
waitOnTaskgridCapLimit	Specifies the name of the data grid to update.
	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
	Specifies the maximum capacity for the selected data grid in megabytes.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.

status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "useLRU": "false",
"gridName": "myGrid", "waitOnTask": "true", "command":
"ModifyGridCapacity", "gridCapLimit": "10" } }
```

Command Type:

grid

ModifyGridReplication

Description:

Changes the replication settings for a data grid.

Required Parameters:

st

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
maximumSyncReplicas	Specifies the maximum number of synchronous replicas for this data grid.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
maximumAsyncReplicas	Specifies the maximum number of asynchronous replicas for this data grid.

Results Parameters:

errorM

essage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"waitOnTask": "true", "maximumSyncReplicas": "4", "command":
"ModifyGridReplication", "maximumAsyncReplicas": "2" } }
```

Command Type:

grid

ModifyGridSecurity

Description:

Changes the security settings for a data grid.

Required Parameter

rs
:

stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
authorizationEnabled	Enables authorization for the data grid when set to true.
gridName	Specifies the name of the data grid to update.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
securityEnabled	Enables security for the data grid when set to true.

Result

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "authorizationEnabled":
"false", "gridName": "myGrid", "waitOnTask": "true", "command":
"ModifyGridSecurity", "securityEnabled": "false" } }
```

Command Type:

grid

ModifyGridTTL

Description:

(Simple data grids only) Changes time to live (TTL) settings for the data grid.

Requi

Required Parameters:

defaultMapEvictorType	(Optional) Specifies the time to live eviction type used for the default map. Valid values: NONE, CREATION_TIME, LAST_ACCESS_TIME, LAST_UPDATE_TIME.
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
waitOnTask	Specifies whether to wait on the completion of the task associated with the command. If the value is set to true, wait on the completion of the task. If the value is set to false, do not wait on the completion of the task.
gridTTL	Specifies the amount of time, in seconds, to keep data before evicting the data from data grid.

Result Para

Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "defaultMapEvictorType": "CREATION_TIME",
"stopOnTaskFailure": "true", "gridName": "myGrid", "waitOnTask":
"true", "command": "ModifyGridTTL", "gridTTL": "60000" } }
```

Command Type:

grid

ModifyGroupAccess

Description:

Changes the group access rights for a data grid.

Required P

Parameters:

groupName	Specifies the name of the group for which you want to modify access rights.
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
accessType	Specifies the type of access to give to the group. Valid values: 1 = read, 2 = write, 3 = create, 4 = all.

Resul

tParameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "groupName": "somegroup",
"gridName": "myGrid", "command": "ModifyGroupAccess",
"accessType": "1" } }
```

CommandType:

grid

ModifyUserAccess

Description:

Changes user access rights for a data grid.

Req

required Parameters:	
stopOnTaskFailure	Specifies whether to stop running the batch routine when the task fails. If the value is set to true, the batch routine stops. If the value is false or blank, the batch routine does not stop. Applies to commands that are run in batch processes only.
gridName	Specifies the name of the data grid to update.
userName	Specifies the name of the user for which access rights are being updated.
accessType	Specifies the type of access to give to the user. Valid values: 1 = read, 2 = write, 3 = create, 4 = all.

Result Parameters:

errorMessage	A message that explains the reason for the failure, if the command failed.
status	Specifies the status of the command that was run. The result can be either success or failed.
commandResult	A JSON-formatted statement that contains the result of the command that was run.
commandName	Specifies the name of the HTTP command interface command that was run.

Example:

```
{ "task": { "stopOnTaskFailure": "true", "gridName": "myGrid",
"command": "ModifyUserAccess", "userName": "someuser",
"accessType": "1" } }
```

Command

Fichier de propriétés du client

Création d'un fichier de propriétés en fonction de vos exigences pour les processus client WebSphere eXtreme Scale.

Exemples de fichier de propriétés du client

Java Vous pouvez utiliser le fichier `sampleClient.properties` qui se trouve dans le répertoire *rép_base_wxs*`/properties` pour créer votre propre fichier de propriétés.

.NET Vous pouvez utiliser le fichier `Client.Net.properties` qui se trouve dans le répertoire *net_client_home*`\config\` pour créer votre propre fichier de propriétés.

Java

Définition d'un fichier de propriétés pour les clients Java

Vous pouvez spécifier le fichier de propriétés du client de l'une des manières ci-après. La spécification d'un paramètre en utilisant l'un des éléments plus loin dans la liste remplace le paramètre précédent. Par exemple, si vous spécifiez une valeur de propriété système pour le fichier de propriétés du serveur, les propriétés contenues dans ce fichier remplacent celles contenues dans le fichier `objectGridClient.properties` qui se trouvent dans le chemin d'accès aux classes.

1. En tant que fichier nommé n'importe où dans le chemin d'accès aux classes. Vous ne pouvez pas placer ce fichier dans le répertoire de travail du système :

   ```
   objectGridClient.properties
   ```

2. En tant que propriété système dans une configuration autonome ou WebSphere Application Server. Cette valeur peut spécifier un fichier du répertoire de travail du système, mais pas un fichier figurant dans le chemin d'accès aux classes :

   ```
   -Dobjectgrid.client.props=nom_fichier
   ```

   Remarque : Dans une configuration WebSphere Application Server, le fichier de propriétés du client doit se trouver dans le chemin d'accès aux classes si vous voulez pouvoir imposer l'utilisation de ce fichier à l'aide de la propriété système ; par exemple, *racine_was*`/properties` ou *racine_profil*`/properties`, selon que vous voulez que le fichier de propriétés s'applique à un profil spécifique ou à toute l'installation.

3. Comme une substitution à l'aide d'un programme, à l'aide de la méthode ClientClusterContext.getClientProperties. Les données de l'objet sont alimentées avec celles des fichiers de propriétés. Vous ne pouvez pas configurer les propriétés de sécurité à l'aide de cette méthode.

.NET

Définition d'un fichier de propriétés pour WebSphere eXtreme Scale Client for .NET

Le fichier par défaut des propriétés de client se trouve dans le répertoire *net_client_home*`\config\Client.Net.properties`. Si vous voulez définir vos propres fichiers de propriétés, fournissez le chemin d'accès complet au fichier de propriétés souhaité dans chaque appel de méthode Connect.

```
IGridManager gm = GridManagerFactory.GetGridManager( );
ICatalogDomainInfo cdi = gm.CatalogDomainManager.CreateCatalogDomainInfo( "localhost:2809"
);
ccc = gm.Connect( cdi, @"C:\MyLocation\MyClient.properties" );
grid = gm.GetGrid( ccc, gridName );
```

Propriétés du client

Propriétés du client

.NET 2.5+ **enableDynamicConfiguration**

Lorsque cette propriété a la valeur `true`, les modifications apportées à la propriété requestRetryTimeout du fichier de propriétés du client sont détectées dynamiquement. La nouvelle valeur de cette propriété est immédiatement utilisée pour calculer le nouveau délai d'attente pour les nouvelles tentatives d'exécution de requête.

Valeur par défaut : `false`

Java listenerHost

Indique le nom d'hôte auquel le protocole de transport ORB (Object Request Broker) ou eXtremeIO (XIO) se lie pour les communications. La valeur doit être un nom qualifié complet de domaine ou une adresse IP. Si la configuration implique plusieurs cartes réseau, configurez l'hôte du programme d'écoute et le port d'écoute pour que le mécanisme de transport dans la machine JVM connaisse l'adresse IP de liaison. Si vous ne définissez pas l'adresse IP à utiliser, des symptômes (dépassements du délai de connexion, défaillances inhabituelles d'API et clients qui semblent se bloquer) apparaissent.

Java listenerPort

Indique le numéro de port auquel le protocole de transport ORB (Object Request Broker) ou eXtremeIO (XIO) se lie pour les communications. Le numéro de port défini pour listenerPort est utilisé pour les communications entre un client et un serveur de catalogue, et entre un serveur de conteneur et un serveur de catalogue qui se trouvent dans le même domaine. Il est également utilisé pour les communications interdomaine et intradomaine entre les serveurs de catalogue.

Valeur par défaut :`2809`

Remarque : Lorsqu'un serveur de grille de données s'exécute dans WebSphere Application Server et que le protocole de transport ORB est utilisé, un autre port ORB_LISTENER_ADDRESS doit également être ouvert. Le port BOOTSTRAP_ADDRESS réachemine les requêtes vers ce port. Si vous utilisez le protocole de transport XIO, le port XIO_ADDRESS doit être ouvert.

Java preferLocalProcess

Cette propriété n'est pas utilisée actuellement. Elle est réservée à une utilisation future.

Java preferLocalHost

Cette propriété n'est pas utilisée actuellement. Elle est réservée à une utilisation future.

Java .NET preferZones

Indique une liste de zones de routage recommandées. Chaque zone spécifiée doit être séparée de la précédente par une virgule, selon le format suivant : `preferZones=ZoneA,ZoneB,ZoneC`

Valeur par défaut : aucune

Java .NET requestRetryTimeout

Indique pendant combien de temps (en millisecondes) le traitement d'une requête doit se poursuivre après qu'une exception s'est produite. Utilisez l'une des valeurs admises suivantes :

- Une valeur égale à `0` indique que la requête doit échouer immédiatement et ignorer la logique interne régissant les nouvelles tentatives.
- Une valeur égale à `-1` indique que le délai entre les tentatives d'exécution de la requête n'est pas défini, ce qui signifie que la durée pendant laquelle le système tente d'exécuter la requête dépend du délai d'expiration des transactions. (Valeur par défaut)
- Une valeur supérieure à `0` indique la valeur du délai d'expiration de la requête en millisecondes. Les exceptions dont la création échoue sont renvoyées. Même lorsque des exceptions, telles que DuplicateException, sont réexécutées, elles sont également renvoyées quand elles échouent. Le délai de transaction reste utilisé comme délai d'attente maximal.

Java .NET shuffleBootstrapAddresses

Indique si les adresses de service de catalogue doivent subir un traitement aléatoire lorsqu'elles sont utilisées par un client qui s'amorce dans la grille de données. La valeur doit être `true` ou `false`.

Valeur par défaut : `true`

Java 2.5+ xioTimeout

Indique le délai d'attente maximum (en secondes) pour les tentatives d'établissement d'une connexion socket sortante par eXtremeIO. Cette valeur est également utilisée comme délai d'attente par défaut par la logique eXtremeIO interne.

Java 2.5+ xioRequestTimeout

Indique pendant combien de secondes une requête attend une réponse avant d'expirer. Cette propriété influence la durée de la reprise en ligne du client en cas d'indisponibilité du réseau. Si vous spécifiez une valeur trop faible pour cette propriété, les demandes risquent d'arriver à expiration trop tôt. Définissez soigneusement la valeur de cette propriété pour éviter les dépassements de délai d'attente inutiles.

Propriétés de sécurité du client

Propriétés de sécurité générales

Java .NET securityEnabled

Active la sécurité du client WebSphere eXtreme Scale. Ce paramètre doit correspondre au paramètre securityEnabled dans le fichier de propriétés de serveur WebSphere eXtreme Scale. Si les paramètres ne correspondent pas, la connexion à la grille de données échoue.

Valeur par défaut : `false`

Propriétés de configuration de l'authentification des données d'identification

Java .NET credentialAuthentication

Indique la prise en charge de l'authentification des données d'identification du client. Utilisez l'une des valeurs admises suivantes :

- `Jamais` : Le client ne prend pas en charge l'authentification des données d'identification.
- `Pris en charge` : Le client prend en charge l'authentification des données d'identification s'il en est de même pour le serveur. (Valeur par défaut)
- `Obligatoire` : Le client requiert l'authentification des données d'identification.

Java .NET authenticationRetryCount

Indique le nombre de tentatives d'authentification si les données d'identification ont expiré. Si la valeur est 0, aucune nouvelle tentative d'authentification n'est exécutée.

Valeur par défaut : 3

.NET credentialGeneratorAssembly

Indique le nom de l'assemblage utilisé pour générer les données d'identification pour le WebSphere eXtreme Scale Client for .NET. Pour que vous puissiez définir cette propriété, la propriété credentialAuthentication doit avoir la valeur `Pris en charge` ou `Obligatoire`. La valeur indiquée doit être un nom d'assemblage `.dll` C# valide avec version, culture et autres propriétés.

Exemple : `IBM.WebSphere.Caching.CredentialGenerator, Version=8.6.0.0, Culture=neutral, PublicKeyToken=b439a24ee43b0816`

Java .NET credentialGeneratorClass

Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Pour que vous puissiez définir cette propriété, la propriété credentialAuthentication doit avoir la valeur `Pris en charge` ou `Obligatoire`. Cette classe est utilisée pour obtenir les données d'identification des clients.

Valeur par défaut : aucune

Java .NET credentialGeneratorProps

Spécifie les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés sont affectées à l'objet à l'aide de la méthode setProperties(String). Pour que vous puissiez définir cette propriété, la propriété credentialAuthentication doit avoir la valeur `Pris en charge` ou `Obligatoire`. La valeur credentialGeneratorprops n'est utilisée que si la valeur de la propriété credentialGeneratorClass n'est pas NULL.

Propriétés de configuration de la sécurité de la couche de transport

Java .NET transportType

Indique le type de transport du client. Les valeurs possibles sont :

- `TCP/IP` : Indique que le client ne prend en charge que les connexions TCP/IP.
- `SSL pris en charge` : Indique que le client prend en charge les connexions TCP/IP et SSL (Secure Sockets Layer). (Valeur par défaut)
- `SSL requis` : Indique que le client requiert des connexions SSL.

Propriétés de configuration SSL

Java alias

Indique le nom d'alias dans le fichier de clés. Cette propriété est utilisée si le fichier de clés contient plusieurs certificats de paire de clés et que vous souhaitez sélectionner l'un des certificats.

Valeur par défaut : aucune

Java contextProvider

Indique le nom du fournisseur de contexte du service sécurisé. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que le type du fournisseur de contexte est incorrect.

Les valeurs valides sont : `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

Java **keyStore**

Indique un chemin complet vers le fichier de clés.

Exemple :

```
etc/test/security/client.private
```

Java **keyStoreType**

Indique le type de fichier de clés. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : JKS, JCEK, PKCS12, etc.

Java .NET **protocol**

Indique le type du protocole de sécurité à utiliser pour le client. Définissez cette valeur de protocole en fonction du fournisseur de sécurité. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que la valeur du protocole est incorrecte.

Java Les valeurs valides sont : SSL, SSLv3, TLS, TLSv1, etc.

.NET Les valeurs valides sont : SSLv2, SSLv3, TLS ou `Par défaut` (SSLv3 ou TLS1.0)

.NET **publicKeyFile**

Définit le nom de chemin complet d'un fichier qui contient la clé publique exportée depuis le serveur.

Exemple : `c:\tmp\wxs\serverA.cer`

Java **trustStoreType**

Indique le type de fichier de clés certifiées. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : JKS, JCEK, PKCS12, etc.

Java **trustStore**

Indique un chemin complet vers le fichier de clés.

Exemple :

```
etc/test/security/server.public
```

Java **keyStorePassword**

Indique le mot de passe de chaîne du fichier de clés. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

Java **trustStorePassword**

Indique un mot de passe de chaîne au fichier de clés certifiées. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

Rubrique parent : Référence

Concepts associés:
IBM WebSphere DataPower XC10 Appliance : présentation de la sécurité

Tâches associées:
Configuration de la sécurité du client
.NET Programmation de l'authentification du client pour WebSphere eXtreme Scale Client for .NET

Information associée:
Interface ICredential
Interface ICredentialGenerator

Syntaxe d'expression régulière

Une expression régulière est une chaîne structurée utilisée pour la mise en correspondances d'autres chaînes. Vous pouvez utiliser des expressions régulières pour les données d'invalidation et pour le filtrage des messages qui s'affichent dans Message Center.

Exemples d'expression régulière

Expression régulière	Description
`.`	Recherche n'importe quel caractère.
`.*`	Recherche n'importe quel caractère, zéro ou plusieurs fois.
`A.*`	Recherche toutes les chaînes qui commencent par "A".
`.*A`	Recherche toutes les chaînes qui se terminent par "A".
`A?`	Recherche "A", zéro ou une fois.
`A*`	Recherche "A", zéro ou plusieurs fois.
`A+`	Recherche "A", une ou plusieurs fois.
`A?B*C+`	Recherche n'importe quel caractère, suivi de zéro ou un A, suivi de zéro ou plusieurs B, suivi d'un ou plusieurs C
`[ABC].*`	Recherche les chaînes commençant par les lettres "A", "B" ou "C".
`.*ABC.*`	Recherche les chaînes comportant la chaîne "ABC" dans la chaîne.
`(ABC|DEF).*`	Recherche les chaînes commençant par la chaîne "ABC" ou "DEF".

Pour la syntaxe complète d'une expression régulière, consultez la documentation de l'API java.util.regex.Pattern.

Rubrique parent : Référence

Options de configuration de mappe dynamique

Vous pouvez créer des mappes supplémentaires dans une grille de données en demandant à votre application client de se connecter à la mappe spécifiquement nommée. Une fois cette connexion établie, la mappe est automatiquement créée.

Attribution de nom à une mappe dynamique

Lorsque vous créez une grille de données, deux mappes sont créées par défaut. La première prend le nom de la grille de données. Par exemple, si vous créez la grille de données `myGrid`, vous obtenez automatiquement une mappe `myGrid`. La seconde est destinée au cache local et prend le nom `myGrid.NCI`. Cependant, vous pouvez également ajouter des mappes à la grille de données. Une mappe est automatiquement créée lorsque l'application client se connecte à une mappe à l'aide de la convention d'attribution de nom suivante :

```
<nom_mappe>.<modèle>.<option_verrouillage>.<invalidation_cache_local>
```

Où :

***nom_mappe* (obligatoire)**

Indique le nom de la mappe.

***modèle* (obligatoire)**

Indique le modèle qui définit la date d'expiration des entrées de la mappe, en définissant le comportement TTL (Time To Live). Pour obtenir la liste des options disponibles, voir Modèles de mappes.

option_verrouillage

Spécifie le mécanisme de verrouillage utilisé pour la mappe. Pour obtenir la liste des options disponibles, voir Options de verrouillage.

2.5+ ***invalidation_cache_local***

Spécifie si l'invalidation de cache local est utilisé pour la mappe. L'invalidation de cache local est utilisée uniquement lorsque la mappe comporte un type de verrouillage défini sur `NONE` ou `OPTIMISTIC`. Vous pouvez configurer l'invalidation du cache local pour supprimer les données périmées du cache local aussi rapidement que possible. Lorsqu'une mise à jour, une suppression ou une invalidation est exécutée sur la grille de données distante, une invalidation asynchrone est déclenchée dans le cache local. Voir Options d'invalidation du cache local pour obtenir la liste options disponibles.

Vous devez inclure un nom de modèle pour la mappe. Si vous indiquez pas d'option de verrouillage, aucun verrouillage n'a lieu sur la mappe.

Modèles de mappes

Tableau 1. Modèles de mappes dynamiques

Modèle de mappe	Description
*.NONE	Spécifie une mappe sans expiration de durée de vie.
*.LUT	Spécifie une mappe dans laquelle les entrées sont arrivées à expiration sur la base de la date/heure de dernière mise à jour de l'entrée. La durée de vie par défaut est d'une heure.
*.LAT	Une mappe dans laquelle les entrées sont arrivées à expiration sur la base de la date/heure de dernier accès à l'entrée. La durée de vie par défaut est d'une heure.
*.CT	Une mappe dans laquelle les entrées sont arrivées à expiration sur la base de la date/heure de création de l'entrée, plus la valeur TTL. La durée de vie par défaut est d'une heure.

Options de verrouillage

Tableau 2. Options de verrouillage de mappe dynamique

Option de verrouillage	Description
(vide)	Si vous indiquez pas d'option de verrouillage, aucun mécanisme de verrouillage n'est utilisé.
.P	Indique que la mappe possède le verrouillage pessimiste
.O	Indique que la mappe possède le verrouillage optimiste

2.5+

Options d'invalidation du cache local

Tableau 3. Options d'invalidation du cache local

Option de verrouillage	Description
(vide)	Si vous n'indiquez aucune valeur, l'invalidation du cache local est désactivée.
.NCI	Indique que la mappe utilise l'invalidation du cache local. Votre mappe doit être configurée avec le verrouillage optimiste ou sans aucun verrouillage.

Rubrique parent : Référence

Concepts associés:
Expulseurs

Tâches associées:
Configuration d'un expulseur TTL (Time To Live)
Configuration des mappes dynamiques
Java 2.5+ Configuration de l'invalidation du cache local
Développement d'applications de grille de données avec la passerelle REST
Création de grilles de données simples
.NET Création de mappes dynamiques avec les API .NET

CLI command reference

This page describes the command line interface (CLI) commands provided by WebSphere® DataPower® XC10 Appliance. The following sets of commands are supported:

- Common commands that work with various configuration objects. See Common commands.
- Special purpose commands. See Special purpose commands.

Common commands

Commands described in this section are used to work with various configuration objects. Common commands either work on a specified configuration object or are used within an edit session on a configuration object. An edit session is a mode of the CLI in which you can modify the values of a configuration object. An edit session starts with the "edit" or "create" commands, and ends with the "exit" or "cancel" commands. Some of these commands, such as "append" or "set", are available only within an edit session.

Currently supported configuration objects include:

- aggregate-interface
- ethernet-interface
- ldap-auth
- logging-settings
- netsec-settings
- network-settings
- vlan-interface

Commands that work within an edit session include the following commands:

- cancel
- create
- delete
- disable
- domain list
- domain show
- domain switch
- edit
- enable
- exit
- list
- reset
- show
- show properties
- show types
- status

Parent topic: CLI command reference

cancel

Purpose

Within an edit session, discard all changes that were made to a configuration object during the edit session or during a nested edit session.

Syntax

cancel

Parameters

None.

Usage Notes

In a nested edit session, cancel discards only the changes that were made in the edit submode.

Related Commands

See exit .

Example

Cancel changes that were made during an edit session on the network-settings configuration object.

```
Console> edit network-settings
Console network-settings> set tcp-retries 10
Console network-settings> cancel
Cancelled
Console>
```

Parent topic: Common commands

create

Purpose

Enter an edit session to create an instance of a configuration object.

Syntax

create *object-type object-instance*

Parameters

object-type
: Type of the object to be created

object-instance
: Name of the object instance to be created

Usage Notes

- Some objects cannot be created in the Command Line Interface. These objects include the following:
  - Objects for which there is only one instance, called singleton objects
  - Objects for which there is a predefined, finite set, such as the `ethernet-interface` objects.
- By default, the AdminState of a newly created object is enabled. .

Related Commands

See delete, edit, and list .

Example

Create a vlan-interface named v1000.

```
Console> create vlan-interface v1000
Console vlan-interface:v1000>
...
```

Parent topic: Common commands

delete

Purpose

Delete an instance of a configuration object.

Syntax

delete *object-type object-instance*

Parameters

object-type
: Type of the object to be deleted

object-instance
: Name of the object instance to be deleted

Usage Notes

Some objects cannot be deleted. These objects include the following:

- Objects for which there is only one instance, called singleton objects
- Objects for which there is a predefined, finite set, such as the `ethernet-interface` objects.

Related Commands

See create, edit, and list.

Example

Delete a vlan-interface named v1000.

```
Console> delete vlan-interface v1000
Console>
```

Parent topic: Common commands

disable

Purpose

- Disable a configuration object or an instance of a configuration object.
- This command changes the administrative state (AdminState) to "disabled". If no errors occur, the operational state (OpState) changes to "down".
- No other configuration values are changed as a result of this command.

Syntax

disable *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be disabled

object-instance
: Name of the object instance to be disabled, if the configuration object is not a singleton object

Usage Notes

Configuration changes can be made to an object that is administratively disabled, but the changes will not be applied until the object is enabled.

Related Commands

See enable.

Example

Disable a vlan-interface named v1000.

```
Console> disable vlan-interface v1000
Console>
```

Parent topic: Common commands

domain list

Purpose

Display the supported domains.

Syntax

domain list

Parameters

None.

Usage Notes

Currently only one domain, "default", is supported.

Related Commands

See domain show and domain switch.

Example

Show currently supported domains.

```
Console> domain list
default
Console>
```

Parent topic: Common commands

domain show

Purpose

Display the currently active domain.

Syntax

domain show

Parameters

None.

Usage Notes

Currently only one domain, "default", is supported.

Related Commands

See domain list and domain switch.

Example

Show currently active domain.

```
Console> domain show
In domain "default"
Console>
```

Parent topic: Common commands

domain switch

Purpose

Change the currently active domain.

Syntax

domain switch *domain*

Parameters

domain
: Required active domain

Usage Notes

Currently only one domain, "default", is supported.

Related Commands

See domain list and domain show.

Example

Change the currently active domain.

```
Console> domain switch default
Switched to "default" domain
Console>
```

Parent topic: Common commands

edit

Purpose

Enter a configuration mode to modify a configuration object or an instance of a configuration object.

Syntax

edit *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be edited

object-instance
: Name of the object instance to be edited, if the configuration object is not a singleton object

Usage Notes

The edit command puts you into an edit session for the configured object. Within this edit session, edit subcommands allow you to modify values for the properties of the configured object.

Related Commands

See create, delete, and list.

Example

Edit ethernet-interface eth3 to set the MTU to 4096.

```
Console> edit ethernet-interface eth3
Console ethernet-interface:eth3> set mtu 4096
Console ethernet-interface:eth3> exit
Console>
```

Parent topic: Common commands

enable

Purpose

- Enable a configuration object or an instance of a configuration object.
- This command changes the administrative state (AdminState) to "enabled". If no errors occur, the operational state (OpState) changes to "up".
- No other configuration values are changed as a result of this command.

Syntax

enable *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be enabled

object-instance
: Name of the object instance to be enabled, if the configuration object is not a singleton object

Related Commands

See disable .

Example

Enable a vlan-interface named v1000.

```
Console> enable vlan-interface v1000
Console>
```

Parent topic: Common commands

exit

Purpose

Leave edit mode or an edit submode.

Syntax

exit

Parameters

None.

Usage Notes

Configuration changes are saved when the edit session is ended, not on leaving an edit submode. This might cause validation errors to be displayed later than expected.

Related Commands

See cancel .

Example

Enter an edit session for an `ethernet-interface` object and enter the ip submode. Add an IP address and then exit the ip submode and exit the edit session.

```
Console> edit ethernet-interface eth2
Console ethernet-interface:eth2> ip
Entering "ip" mode
Console ethernet-interface:eth2 ip> append address 192.168.0.200/24
Console ethernet-interface:eth2 ip> exit
Console ethernet-interface:eth2> exit
Console>
```

Parent topic: Common commands

list

Purpose

Show the names of configured objects of a specified object type.

Syntax

list *object-type*

Parameters

object-type
: Type of the object to be listed

Usage Notes

Use show types to get the list of object types.

Related Commands

See create, delete, edit, and show types.

Example

List all the instances of ethernet-interfaces.

```
Console> list ethernet-interface
        eth0
        eth1
        eth2
        eth3
Console>
```

Parent topic: Common commands

reset

Purpose

Restore a configuration object or a part of a configuration object to its default settings.

Syntax

- Reset a singleton configuration object: reset *object-type*
- Reset an instance of a configuration object: reset *object-type object-instance*
- Reset a field of a configuration object in an edit session: reset *property*

Parameters

object-type
: Type of the object to be reset

object-instance
: Instance of the object to be reset, if the configuration object is not a singleton object

property
: Field within a configuration object to be reset

Example

- Reset the IP addresses for ethernet-interface eth3.

```
Console> edit ethernet-interface eth3
Console ethernet-interface:eth3> ip
Entering "ip" mode
Console ethernet-interface:eth3 ip> reset address
Console ethernet-interface:eth3 ip> exit
Console ethernet-interface:eth3> exit
Console>
```

- Reset the entire configuration object for ethernet-interface eth3 to default values.

```
Console> reset ethernet-interface eth3
Console>
```

Parent topic: Common commands

show

Note: The show command is used to display both configuration objects and other specific data. This section documents the use of the show command for configuration objects. See also the show commands within the Special purpose commands section.

Purpose

Show the configured values for a specified configuration object or an instance of a configuration object.

Syntax

show [*object type* [*object instance*]]

Parameters

object type
: Type of the object to be displayed

object instance
: Name of the object instance to be displayed, if the configuration object is not a singleton object

Usage Notes

- Within an edit session, the show command displays the currently configured values for the object instance that is being edited.
- The operational state of the object is also displayed. Only properties that are required and properties that have had values configured differently from the assigned default values are displayed. Optional properties that have not been explicitly configured will not be displayed.
- The values displayed here are the desired configuration for the object. In contrast, the status command will display the actual runtime values for the object.

Example

Show the configured values for the network-settings (singleton) object.

```
Console> show network-settings
network-settings : [Up]

name "network-settings"
AdminState "Enabled"
icmp-options " "
explicit-congestion-notification "false"
destination-based-routing "false"
interface-isolation "relaxed"
tcp-retries "5"
arp-retries "8"
arp-retry-interval "500"
tso-offload "true"
reverse-path-filtering "false"
tcp-window-scaling "true"
```

Parent topic: Common commands

show properties

Purpose

Within an edit session, display the configurable properties of the object.

Syntax

show properties

Parameters

None.

Usage Notes

To see the properties of a nested level, enter the edit submode and repeat the command.

Example

Display the configurable properties of an ethernet-interface.

```
Console> edit ethernet-interface eth3
Console ethernet-interface:eth3> show properties
Basic properties:
        AdminState
        mac-address
        mode
        mtu
        use-arp
        userdata

Sub-mode (complex) properties:
        ip

Console ethernet-interface:eth3> ip
Entering "ip" mode
Console ethernet-interface:eth3 ip> show properties
Basic properties:
        dad-retransmit-timer
        dad-transmits
        use-dhcp
        use-slaac

Array (list) properties:
        address
        static-route

Sub-mode (complex) properties:
        ipv4-default-gateway
        ipv6-default-gateway

Console ethernet-interface:eth3 ip>
```

Parent topic: Common commands

show types

Purpose

Display the types of configuration objects that are supported on the appliance. This does not reflect any instances that are configured, but just types that are supported.

Syntax

show types

Parameters

None.

Usage Notes

None.

Example

Display the configuration objects on the appliance.

```
Console> show types
The following types are defined:
        aggregate-interface
        ethernet-interface
        logging-settings
        network-settings
        vlan-interface
```

Parent topic: Common commands

status

Note: The status command is used to display both status of configured objects and specific functional status information. This section documents the use of the status command for configuration objects. See also the status commands within the Special purpose commands section.

Purpose

Show the status of a specified configuration object or an instance of a configuration object.

Syntax

status *object-type* [*object-instance*]

Parameters

object-type
: Type of the object to be displayed

object-instance
: Name of the object instance to be displayed, if the configuration object is not a singleton object

Usage Notes

- If the *object-instance* is omitted for a configuration object that is not a singleton object, the status of all instances of the specified *object-type* is displayed.
- The status command is supported only for selected configuration objects.
- The values displayed here are the runtime values for the object. In contrast, the show command will display the configured properties of the object.

Example

Show the status of ethernet-interface eth0.

```
Console> status ethernet-interface eth0
eth0     OpState:[Up]
        generic MTU:1500 carrier:true flags:UP BROADCAST RUNNING MULTICAST
            index:5
        inet addr:9.42.77.48 flags:PERMANENT mask:255.255.255.0
            scope:GLOBAL
        inet6 addr: 2002:92a:8f7a:901:9:42:77:48 flags:PERMANENT
            mask: ffff:ffff:ffff:ffff:: scope:GLOBAL
        inet6 addr: fe80::20c:29ff:fedd:7baf flags:PERMANENT
            mask: ffff:ffff:ffff:ffff:: scope:LINK
        ethernet Link:on MAC: 00:0c:29:dd:7b:af autoneg:on duplex:Full
            port:TP speed:1000Mbps
        statistics collisions:0 multicast:0 rx_bytes:69088545
            rx_compressed:0 rx_crc_errors:0 rx_dropped:0 rx_errors:0
            rx_fifo_errors:0 rx_frame_errors:0 rx_length_errors:0
            rx_missed_errors:0 rx_over_errors:0 rx_packets:842919
            tx_aborted_errors:0 tx_bytes:3512656 tx_carrier_errors:0
            tx_compressed:0 tx_dropped:0 tx_errors:0 tx_fifo_errors:0
            tx_heartbeat_errors:0 tx_packets:18144 tx_window_errors:0
```

Parent topic: Common commands

Special purpose commands

Commands described in this section take a specific action on the appliance or display specific information.

- packet-capture start
- packet-capture stop
- platform collect-pd
- platform log-level get
- platform log-level set
- platform service disable ssh
- platform service disable telnet
- platform service enable ssh
- platform service enable telnet
- platform must-gather
- raid delete
- raid re-use
- raid foreign-config import
- raid foreign-config clear
-  request force-suspend
-  request resume
-  request suspend
- set-dns-search
- set-dns-servers
- set-ntp-servers
- show commandref
- show components
- show locales
- show log
- show timezones
- show version
- sshkey fingerprint
- sshkey reset
- sshkey verify
- start-progress
- status battery
- status cpu-usage
- status cpu-utilization
- status dynamic-tunnels
- status fan
- status flash
- status intrusion
- status memory
- status nvdimm
- status power-supply
- status raid all
- status raid battery
- status raid physical
- status voltage
- status volume
- status temperature
- status uptime
- timezone get
- timezone set
- unalias
- usage
- user add
- user delete
- user known-hosts list
- user known-hosts delete
- user list
- user password
- user sshkey add
- user sshkey delete

Parent topic: CLI command reference

add-jvm-args

Purpose

Adds a predetermined set of JVM configuration parameters to any of the processes on the appliance during startup.

Syntax

add-jvm-args process name listOfOneOrMoreJVMArgs

Parameters

There are two required parameters for this command:

process name
: Name of the process that needs to be configured. Valid processName arguments include:

  - all - All processes, including xsa.app, catalog server and all container servers.
  - console - Includes the xsa.app process only.
  - grids - All container server processes only.
  - gridNN - Where NN equals a number (i.e 01) that specifies a single container server.

listOfOneOrMoreJVMArgs
: List valid JVM arguments for the process. Valid JVM arguments you can specify include:

  - -Dcom.ibm.CORBA.Debug
  - -Dcom.ibm.CORBA.CommTrace
  - -Djavax.net.debug
  - -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true
  - -verbose:gc
  - -Xverbosegclog
  - -Xhealthcenter
  - -Xcheck:jni

  Note: If there is more than one parameter you want to include in the list, separate with `!:!`, for example: `add-jvm-args grid01 -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true!:!`

  `-Dcom.ibm.CORBA.CommTrace add-jvm-args all -Dcom.ibm.CORBA.CommTrace`

Related Commands

See device restart.

Example

```
Console> add-jvm-args grid01 -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true
Updating JVM Args to include new args...
Adding the following JVM Arg: -Dcom.ibm.websphere.xs.ProduceJavaCoreOnLockTimeout=true
[Wed Apr 10 2013 17:00:22]  Run device restart to enable the new JVM args settings.
```

Parent topic: Special purpose commands

alias

Purpose

- Define an alternative name or a shortcut for a command line interface (CLI) command.
- Display configured aliases.

Syntax

- Display the list of defined aliases: alias
- Display the definition for an existing alias: alias *name*
- Define an alias for a specified CLI command: alias *name value*

Parameters

name
: Alternative name for a command line interface (CLI) command.

value
: Substitute value for the specified name. If the substitute value contains spaces, enclose the value in double quotation marks.

Related Commands

See unalias.

Example

Define a shortcut for testing connectivity to a particular address. Then display the configured aliases.

```
Console> alias test-connectivity "net-test ping 9.42.106.2"
Console> alias
alias test-connectivity "net-test ping 9.42.106.2"
Console>
```

Parent topic: Special purpose commands

clear-all

Purpose

Resets the configuration data for the appliance, including removing all the data grids and new users. All of the internal processes that are running on the appliance are restarted. This command runs without restarting the appliance hardware.

Syntax

clear-all

Parameters

None.

Related Commands

None.

Example

```
Console> clear-all
Force Stopped all XC-10 processes
Deleting configuration data and logs
Deleting grid data
```

Parent topic: Special purpose commands

clear-jvm-args

Purpose

Clears JVM configuration parameters to any of the processes that had been applied to the appliance during startup.

Syntax

clear-jvm-argsprocessName

Parameters

process name
: Name of the process that needs to be configured. Valid processName arguments include:

  - all - All processes, including xsa.app, catalog server and all container servers.
  - console - Includes the xsa.app process only.
  - grids - All container server processes only.
  - gridNN - Where NN equals a number (i.e 01) that specifies a single container server.

Related Commands

See add-jvm-args.

Example

```
Console> clear-jvm args all
```

Parent topic: Special purpose commands

clear-logs

Purpose

Deletes all of the WebSphere® DataPower® XC10 Appliance log files.

Syntax

clear-logs

Parameters

None.

Related Commands

None.

Example

```
Console> clear-logs
Cleared All Logs
```

Parent topic: Special purpose commands

clear-tls-config

Purpose

Resets the Transport Layer Security (TLS) configuration. Run this command if clear-all is not an option because you do not want to lose all configuration data, but TLS configuration becomes corrupted or you want to restore the default TLS values. In most situations, use the user interface for TLS changes or use the clear-all command to revert to the original configuration of the appliance. Run the clear-tls-config command on each appliance in the collective. After running the command on each appliance, restart the processes in each appliance in the collective. If the collective is successfully communicating, use the device restart command. The collective is communicating properly when all appliances in the collective are accessible through the user interface and can be seen as started in the Collective panel. However, if the TLS configuration is preventing the collective from communicating and the device restart command does not bring the appliance back up, you can use the force-recycle command to forcibly stop and start all the processes in the appliance without saving any data.

Syntax

clear-tls-config

Parameters

None.

Related Commands

clear-all, device restart, and force recycle.

Example

```
Console> clear-tls-config
Cleared Transport Layer Security Configuration
Run clear-tls-config on the other appliances in the collective and restart each appliance for chan
ges to take effect
```

Parent topic: Special purpose commands

collect-logs

This command has been deprecated. Use platform must-gather instead.
Parent topic: Special purpose commands

component firmware update

Purpose

Update component firmware version to configured version if current firmware version is not matched.

Syntax

component firmware update component type

Parameters

component type

- component type name. see help for this command to get a full list of supported component types.
- when using show components and if a component firmware version is not matched with the configured version, you can use this command to update its firmware version.

Related Commands

See show components.

Example

Update component firmware for BMC.

```
Console> component firmware update bmc
CWZBR03003I: Are you sure you want to update component firmware for bmc
After update, device will reboot
CWZBR03012I: yes/no:yes
CWZBR03013I: Component firmware update is starting.....
CWZBR03006I: Component firmware update succeeded and reboot device
```

Parent topic: Special purpose commands

config

Purpose

Import or export appliance configurations.

Syntax

```
config <export|import|usage|listExports> [-file] <filename> [-keyStore] <keystore>
[-trustStore] <truststore> [-silent] [-noRestart]
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

-truststore
: The -trustStore flag is followed by the truststore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-keystore
: The -keyStore flag is followed by the keystore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-noRestart
: Certain appliance settings require you to restart your appliances. If you set this flag, then none of your appliances are restarted. However, if you set the -noRestart parameter, then you must manually restart your appliances to preserve settings that require it.

-silent
: When specified, the config command status messages are not displayed, which is the default behavior.

Usage notes

None

Related commands

This command has the following subcommands: See usage, export, import, listExports, and deleteExport.

Example

Import or export appliance configurations.

```
config export -file foo.json -silent
config import -file foo.json -keyStore keyStore.jks -trustStore trustStore.jks
config import -file foo.json -noRestart
config usage
config listExports
config deleteExport export_1942-09-06_13-50-14.036.json
```

Parent topic: Special purpose commands

datetime get

Purpose

Display the time that is configured on the appliance, both in Greenwich mean time (GMT) and local time.

Syntax

datetime get

Parameters

None.

Related Commands

See datetime set, timezone get, and timezone set.

Example

Display the configured time on the appliance.

```
Console> datetime get
GMT: 2012-03-08T20:20:21Z
Local: Mar 8, 2012 3:20:21 PM
Console>
```

Parent topic: Special purpose commands

datetime set

Purpose

Configure the current time on the appliance.

Syntax

datetime set *time*

Parameters

time

- Current time in local time (as determined by the timezone setting).
- Format is `YYYY-MM-DD hh:mm:ss` or `YYYY-MM-DD hh:mm:ssZ` with the following specifications:

  `YYYY`
  : Represents the year

  `MM`
  : Represents the month

  `DD`
  : Represents the day of the month

  `hh`
  : Represents hour

  `mm`
  : Represents minutes

  `ss`
  : Represents seconds

  `Z`
  : Appends to the time to specify it in Greenwich mean time (GMT).

Related Commands

See datetime get, timezone get, and timezone set.

Example

Display the configured timezone, set the current time on the appliance and then display configured time.

```
Console> timezone get
Timezone is EST
Console> datetime set 2012-03-22 10:01:00
Console> datetime get
GMT: 2012-03-22T15:01:03Z
Local: Mar 22, 2012 10:01:03 PM
Console>
```

Parent topic: Special purpose commands

deleteExport

Purpose

Delete a specified export file from the list of exports that are stored locally.

Syntax

```
config deleteExport
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

Usage notes

You must pass the deleteExport command and specify only one file with the -file flag.

Example

Delete exported appliance configurations.

```
config deleteExport export_1942-09-06_13-50-14.036.json
```

Parent topic: Special purpose commands

device RESET

Purpose

Erase all data on the appliance and put it back to a factory-new state. All configuration is deleted, including the ssh server keys, license acceptance, and logs. All user IDs and passwords are also deleted and replaced with the default user ID and password that were set at the factory. The appliance will automatically reboot after you run this command.

The "RESET" part of the command is required to be all uppercase to lower the probability of it being typed accidentally. This command differs from firmware pristine-install because this command does not change the firmware that is running on this device. Therefore, you do not need a firmware image when you run this command. This command changes only the configuration.

Syntax

device RESET [noprompt]

Options

noprompt
: Do not prompt for confirmation. The command is started immediately. If this option is not present, the command is not started unless the user answers affirmatively to a confirmation prompt that follows.

Parameters

None.

Usage Notes

If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is ended when the configuration is cleared. If you want to see all the messages for this command, you need to be connected to the serial console.

Related Commands

See firmware pristine-install.

Example

```
Console> device RESET

This command will result in the network configuration being reset; this may end your
current connection to the appliance.

Undo all existing configuration and reset the appliance to factory-new condition? y/n: y
You have chosen to reset the device, proceeding with reset...
Reset complete; ready to reboot
Console>
```

Parent topic: Special purpose commands

device battery-replaced

Purpose

Log the date on which the internal battery was replaced. The battery provides backup power for the hardware configuration and RAID disk writeable cache when the appliance is powered off and unplugged from an AC power source.

Batteries are valid for a specified duration, and this command marks the beginning of that duration. This command was run at the factory, and needs to be run by a user only when a battery is replaced in the field. The current installation date and expiration date of the existing battery can be shown with the status battery command.

Syntax

device battery-replaced

Options

None.

Parameters

None.

Related Commands

See status battery.

Example

```
Console> device battery-replaced
Ok
Console>
```

Parent topic: Special purpose commands

device intrusion allow

Purpose

- Allow the case to be opened on those devices that allow it.
- See device intrusion clear.

Syntax

device intrusion allow

Parameters

None.

Example

```
Console> device intrusion allow
Ok
Console>
```

Parent topic: Special purpose commands

device intrusion clear

Purpose

- Clear device of intruded state recorded when the appliance case is physically opened so that the case is no longer considered intruded.
- Some customer-replaceable or field-replaceable parts on the appliance might require the case to be opened. Run this command after the parts are replaced and the case is closed.

Syntax

device intrusion clear

Parameters

None.

Usage Notes

- This command is valid only on a 7198 or 7199 machine types.
- Opening the case on a 9235 machine type will render the appliance permanently inoperable, requiring that the appliance be returned to the factory, so do not open the case on a 9235 machine type. These physical appliances have sensors that detect when the case has been opened. On other machine types of physical appliances, this command will not have any effect. This command is not applicable on virtual appliances.
- See also status intrusion.

Example

```
Console> device intrusion clear
Ok
Console>
```

Parent topic: Special purpose commands

device intrusion disallow

Purpose

Reenable intrusion detection after a previous invocation of device intrusion allow.

Syntax

device intrusion disallow

Parameters

None.

Usage Notes

- This command has effect only on machine types 7198 and 7199. See device intrusion clear for more information.
- This command does not reset the intrusion sensor. It is generally a good idea to invoke device intrusion clear before rebooting the appliance.

Example

```
Console> device intrusion disallow
Ok
Console>
```

Parent topic: Special purpose commands

device nvdimm clear

Purpose

Clear the nvDIMM volume labels.

Syntax

device nvdimm clear

Parameters

None.

Usage Notes

This command is only allowed on 9006 machine types.

Related Commands

None.

Example

```
Console> device nvdimm clear
CWZBR02923I:  Volume labels cleared
Console>
```

Parent topic: Special purpose commands

device clear-intrusion

This command is deprecated. Use device intrusion clear instead.
Parent topic: Special purpose commands

device restart

Purpose

Shut down the appliance and reboot it. On reboot, all configured applications start. This is the preferred method to reboot the appliance instead of power cycling it.

Syntax

device restart

Options

None.

Parameters

None.

Usage Notes

If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is ended during the shutdown. If you want to see all the messages for the full restart cycle, you need to be connected to the serial console.

Related Commands

See device shutdown.

Example

```
Console> device restart
Ok
Console>
Preparing system for shutdown: s1  hb  hn  2  0  s2  rs  nt  hc  ns  sy  ul r6
machine restart
Starting system... dn un nvm 7198 r1 r2 lo wf lu pt tc sr in cd ln mu ku up us
t1 t2 md mm mcg mt ml na fi
login:
```

Parent topic: Special purpose commands

device shutdown

Purpose

Shut down the appliance and leave it in a powered-off state. It does not automatically reboot. If you want to reboot the appliance after running this command, push the power button if it is a single push button, or toggle the power button if it is a 2-position rocker switch. This is the preferred method to shut down the appliance instead of using only the power button.

Syntax

device shutdown

Options

None.

Parameters

None.

Usage Notes

If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is ended during the shutdown. If you want to see all the messages for the full shutdown cycle, you need to be connected to the serial console.

Related Commands

See device restart.

Example

```
Console> device shutdown
Ok
Console>
Preparing system for shutdown:  s1  hb  hn  2  0  s2  rs  nt  hc  ns  sy  ul  r0
machine shutdown
```

Parent topic: Special purpose commands

echo

Purpose

Write to the output stream of the command line interface (CLI).

Syntax

echo *text*

Parameters

text
: Text to be written to the stream of the command line interface (CLI). It is displayed only in the command line interface (CLI) instance from which this command was issued.

Usage Notes

The echo command is useful in command line interface (CLI) scripts.

Example

Use the echo command to prepend output of the datetime get display.

```
Console> echo Current time is: ; datetime get
Current time is:
GMT: 2012-03-22T15:32:06Z
Local: Mar 22, 2012 3:32:06 PM
Console>
```

Parent topic: Special purpose commands

export

Purpose

Export the appliance configuration either locally or to a temporary location for uploading to a remote file server.

Syntax

```
config export
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

Usage notes

You must pass the -file flag on the `export` command.

Example

Export the appliance configuration.

```
config export -file foo.json -silent
```

Parent topic: Special purpose commands

file delete

Purpose

Delete a file from the appliance.

Syntax

file delete *filename*

Parameters

filename
: File to be deleted

Related Commands

See file get, file list, and file put.

Example

Delete a file that is previously created by the platform collect-pd command.

```
Console> file list
collect-pd.txt 103576 bytes created Mar 22, 2012 3:38:32 PM
Console> file delete collect-pd.txt
Console>
```

Parent topic: Special purpose commands

file get

Purpose

Copy a file to the appliance from a remote location.

Syntax

file get *url localfile*

Parameters

url
: Uniform resource locator that is specified in one of the two following formats. In the following formats, white space is added only for readability. Do not included any white space in the URL on the command.

- For protocols HTTP and FTP:

  protocol`: // [`*username* `[:`*password*`] @]`*hostname* `[:`*port*`]/[%2F` *abspath*`/][`*relpath* `/]` *filename*

  where

  protocol
  : FTP or HTTP.

  username
  : User name at the remote host. This is an optional component of URL.

  password
  : Password for *username* at the remote host. This is an optional component of URL.

  hostname
  : Remote host name or IP address.

  port
  : Decimal port number of the remote server. This is an optional component of URL.

  abspath
  : Absolute path for the remote file, prefixed with the 3 characters '%2F' or '%2f'. It is supported for protocol FTP and SCP and is an optional component of URL.

  relpath
  : Relative path for the remote file. This path is relative to the default directory on the remote system. If *username* is specified, the default directory is typically the home directory of that user. This path is an optional component of URL.

  filename
  : Remote file name.

- For protocols SCP:

  protocol`: //` *username*`@`*hostname*`: [%2F` *abspath*`/][`*relpath* `/]` *filename*

  where

  protocol
  : SCP.

  username
  : User name at the remote host. This is an optional component of URL.

  hostname
  : Remote host name or IP address. Host names that resolve to IPv6 addresses are supported; however, literal IPv6 addresses, which contain colons, are not supported because the *hostname* is separated from the file path with a colon.

abspath
: Absolute path for the remote file, prefixed with the 3 characters '%2F' or '%2f'. It is supported for protocol FTP and SCP and is an optional component of URL.

relpath
: Relative path for the remote file. This path is relative to the default directory on the remote system. If *username* is specified, the default directory is typically the home directory of that user. This path is an optional component of URL.

filename
: Remote file name.

localfile
: Name to be assigned to the local copy of the file. No absolute path elements (leading forward slashes) or subdirectories (forward slashes) are allowed. It must be a relative, flat name.

Related Commands

See file delete, file list, and file put.

Example

Retrieve a firmware file from a remote location and give it the name of `newfirmware` on the appliance.

```
Console> file get scp://user1@server1.raleigh.ibm.com:~/dev_bedrock.scrypt2 newfirmware.scrypt2
user1@server1.raleigh.ibm.com's password:
dev_bedrock.scrypt2                           100%  136MB  34.0MB/s   00:04
Wrote 142553856 bytes to local storage
Console>
```

Parent topic: Special purpose commands

file list

Purpose

List the files on the appliance.

Syntax

file list

Parameters

None.

Related Commands

See file delete, file get, and file put.

Example

Show the files that are currently on the appliance.

```
Console> file list
collect-pd.txt 103794 bytes created Mar 22, 2012 5:30:56 PM
newfirmware 142553856 bytes created Mar 22, 2012 4:41:17 PM
Console>
```

Parent topic: Special purpose commands

file put

Purpose

Copy a file to a remote location from the appliance.

Syntax

file put *localfile url*

Parameters

localfile
: Name of the local file to be transferred. This file must exist.

url
: Uniform resource locator, as described in file get; however, the HTTP protocol is not supported for the file put command.

Related Commands

See file delete, file get, and file list.

Example

Copy local file `collect-pd.txt` to a remote system as `diag.txt`.

```
Console> file put collect-pd.txt scp://user1@system1.rtp.raleigh.ibm.com:~/diag.txt
user1@system1.rtp.raleigh.ibm.com's password:
collect-pd.txt                                100%  101KB 101.4KB/s   00:00
Console>
```

Parent topic: Special purpose commands

firmware pristine-install

Purpose

Install firmware and restore the appliance to a near-factory-fresh state.

Syntax

firmware pristine-install *image*

Parameters

image
: Firmware image to be installed, in the form of an scrypt2 file

Usage Notes

- The pristine-install command installs the new firmware image without copying over configuration or data files copied over on a typical firmware upgrade.
- The pristine-install command can be used to install an older version of firmware on the appliance.
- The pristine-install command can also be used to attempt recovery in error situations where less disruptive methods do not succeed.
- The appliance is rebooted as part of this operation, so network connectivity is disrupted. Therefore, it is recommended that this command be typically run from the serial console.

Related Commands

See firmware upgrade, firmware rollback, and device RESET.

Example

Revert the firmware to an older firmware level.

```
Console> firmware pristine-install oldfirmware.scrypt2
Upgrading firmware...
Verifying image signature
Executing dynamic loader
Executing dynamic loader
Validating image
Extracting firmware from image
Extracting firmware manifest
Executing pre-installation
Deleting previous installation
Linking common files.
Extracting files
Verifying installation
Copying configuration from existing installation
Switching to new installation
Upgrade or rollback succeeded. Rebooting
```

Parent topic: Special purpose commands

firmware rollback

Purpose

Revert the firmware level to the previous level of firmware that is installed on the appliance.

Syntax

firmware rollback

Parameters

None.

Usage Notes

- The appliance holds at most two levels of firmware, the active one and the alternate. Successive invocations of the firmware rollback command result in switching back and forth between the two images.
- This command rolls back the configuration. After the rollback, the configuration is in the state it was in before the last upgrade.
- This command rolls back the component firmware. Because each component can have only one copy of component firmware, the component firmware from the version that is rolled back is reinstalled to the component.
- The appliance is rebooted as part of this operation, so network connectivity is disrupted. Therefore, it is recommended that this command be typically run from the serial console.

Related Commands

See firmware upgrade, firmware pristine-install, and device RESET.

Example

Revert the appliance to the previous version of firmware that is installed.

```
Console> firmware rollback
Rolling back firmware...
Upgrade or rollback succeeded. Rebooting...
```

Parent topic: Special purpose commands

firmware upgrade

Purpose

Install new level of firmware on the appliance.

Syntax

firmware upgrade *image*

Parameters

image
: Firmware image to be installed, in the form of an scrypt2 file

Usage Notes

The appliance is rebooted as part of this operation, so network connectivity is disrupted. Therefore, it is recommended that this command be typically run from the serial console.

Related Commands

See firmware pristine-install, firmware rollback, and device RESET.

Example

Install a new level of firmware on the appliance.

```
Console> firmware upgrade myfirmware.scrypt2
Upgrading firmware...
Verifying image signature
Executing dynamic loader
Validating image
Extracting firmware from image
Extracting firmware manifest
Executing pre-installation
Deleting previous installation
Linking common files
Extracting files
Verifying installation
Copying configuration from existing installation
Switching to new installation
Upgrade or rollback succeeded. Rebooting
```

Parent topic: Special purpose commands

force recycle

Purpose

Restarts the WebSphere DataPower XC10 Appliance processes without saving any data. Because data loss can occur, run this command only if you are not worried about data loss or you have tried the device restart command and the appliance did not become available.

Syntax

force recycle

Parameters

None.

Related Commands

device restart

Example

```
Console> force-recycle
Force Stopped all XC-10 processes
Forced recycle of appliance
```

Parent topic: Special purpose commands

get-dns-search

Purpose

Display configured domain name server search domains.

Syntax

get-dns-search

Parameters

None.

Related Commands

See set-dns-search, get-dns-servers, and set-dns-servers.

Example

Configure and then display domain name server search domains.

```
Console> set-dns-search short.example.com example.com
Ok
Console> get-dns-search
DNS search domains:
        short.example.com
        example.com
Console>
```

Parent topic: Special purpose commands

get-dns-servers

Purpose

Display configured domain name server IP addresses.

Syntax

get-dns-servers

Parameters

None.

Related Commands

See set-dns-servers, get-dns-search, and set-dns-search.

Example

Configure and then display domain name servers.

```
Console> set-dns-servers 9.42.106.2  2002:92a:8f7a:106:9:42:106:42
Ok
Console> get-dns-servers
Domain (DNS) servers:
        9.42.106.2
        2002:92a:8f7a:106:9:42:106:42
Console>
```

Parent topic: Special purpose commands

get-ntp-servers

Purpose

Display configured network time protocol servers.

Syntax

get-ntp-servers

Parameters

None.

Related Commands

See set-ntp-servers.

Example

Configure and then display NTP servers.

```
Console> set-ntp-servers example.server.com 127.0.0.1
Ok
Console> get-ntp-servers
NTP servers:    example.server.com
                127.0.0.1
Console>
```

Parent topic: Special purpose commands

help

Purpose

Display command help information.

Syntax

- Display list of supported commands: help command
- Display help for a specific command: help *command*

Parameters

command
: Command line interface (CLI) command

Example

Display subcommands for the platform command.

```
Console> help platform
The following platform commands are available:
        platform collect-pd [PDFfilename | console]
        platform log-level ...
        platform must-gather filename [PDFfilename]
```

Parent topic: Special purpose commands

import

Purpose

Import the appliance configuration.

Syntax

```
config import
```

Parameters

-file
: The -file flag is followed by the file name where you want to store configuration content.

-truststore
: The -trustStore flag is followed by the truststore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-keystore
: The -keyStore flag is followed by the keystore file name that is a part of your appliance configuration. When you import SSL settings other than those settings that are stored in the configuration list, such as the export_TIMESTAMP.json setting, you must supply this value.

-noRestart
: Certain appliance settings require you to restart your appliances. If you set this flag, then none of your appliances are restarted. However, if you set the -noRestart parameter, then you must manually restart your appliances to preserve settings that require it.

-silent
: When specified, the config command status messages are not displayed, which is the default behavior.

Usage notes

You must pass the import command with the -file flag.

Example

Import the appliance configuration.

```
config import -file foo.json -silent
```

Parent topic: Special purpose commands

license accept

Purpose

Display licenses and require the command line interface (CLI) user to accept or reject licenses.

Syntax

license accept

Parameters

None.

Example

Run the license acceptance process.

```
Console> license accept
Bedrock Project, WebSphere Technical Institute
Copyright 2009-2010, IBM Corporation

This is your sample license.

Accept; Reject: accept
Console>
```

Parent topic: Special purpose commands

license get

Purpose

Transfer a new or changed license to the appliance.

Syntax

license get *url*

Parameters

url
: Remote location of license to be retrieved

Usage Notes

URL must be specified with one of the following protocols: HTTP, FTP, or SCP. See URL syntax for a description of the URL syntax.

Example

Retrieve new license `lic-test` to license repository.

```
Console> license get scp://user1@server1.raleigh.ibm.com:~/temp/lic
user1@server1.raleigh.ibm.com's password:
lic                                           100%   14     0.0KB/s   00:00
Wrote 14 bytes to local storage
Console>
```

Parent topic: Special purpose commands

license reset

Purpose

Reset the license acceptance status.

Syntax

license reset

Parameters

None.

Usage Notes

The next invocation of the command line interface (CLI) requires that the user accept licenses.

Example

Reset the license acceptance status.

```
Console> license reset
Ok
```

Parent topic: Special purpose commands

listExports

Purpose

Display a list of exported configurations that are stored locally.

Syntax

```
config listExports
```

Parameters

None

Usage notes

None

Example

List exported appliance configurations.

```
config listExports
```

Parent topic: Special purpose commands

locale get

Purpose

Retrieve the locale setting used to control display of messages in the command line interface (CLI).

Syntax

locale get

Parameters

None.

Example

Retrieve the locale setting.

```
Console> locale get
Locale is en_US
Console>
```

Parent topic: Special purpose commands

locale set

Purpose

Set the locale to be used in display of messages in the command line interface (CLI).

Syntax

locale set *language* [*country*]

Parameters

language
: Abbreviation for language

country
: Abbreviation for country

Usage Notes

The locale setting will take effect immediately for the CLI in which the command was issued. For other CLI instances that might already be active, the user must exit and reenter the CLI for the new locale setting to take effect.

Example

Set the locale to English by using the conventions of the United States.

```
Console> locale set en US
Console>
```

Parent topic: Special purpose commands

locate-led

Purpose

Turn on or off the locate LED. The locate LED is a light on the front of the physical appliance. It is labeled with an icon in the shape of a lighthouse, and the LED color is blue. The intent of the locate LED is to help customers visually locate a particular appliance in a datacenter. The locate LED is off by default, and must be explicitly turned on by using this command. The state of the LED is not persisted, so after an appliance is rebooted, its locate LED will be off. The locate LED has no function other than as a manual visual locator. This applies only to physical appliances. A virtual appliance has no locate LED.

Syntax

locate-led { on | off}

Options

None.

Parameters

on
: Turn on the locate LED

off
: Turn off the locate LED

Usage Notes

It is not possible to query the state of the locate LED. Setting the state of the locate LED to a state it is in is not considered an error.

Example

```
Console> locate-led on
Ok
Console>
```

Parent topic: Special purpose commands

log message

Purpose

- Write a text string into the diagnostic log file.
- For example, use this command to indicate the start or end of a particular activity. A manual review or automated post-processing of the log file can find the text and use it as a starting point for further analysis.
- This message is logged at the INFO (informational) level.

Syntax

log message *text*

Parameters

text
: Message text to write to the log

Example

```
Console> log message About to run the acceptance tests
Console>
```

Parent topic: Special purpose commands

netif

The netif commands are deprecated and are no longer present by default.

Use the commands documented in Common commands on the `ethernet-interface` configuration object instead.

Parent topic: Special purpose commands

net-test available

Purpose

Test whether a network interface has an active carrier.

Syntax

net-test available

Parameters

None.

Related Commands

For other diagnostic commands, see net-test dns, net-test ping, and net-test tcp.

Example

Test availability of an active network interface.

```
Console> net-test available
Network available
Console>
```

Parent topic: Special purpose commands

net-test dns

Purpose

Perform domain name server lookup on specified host name.

Syntax

net-test dns *hostname*

Parameters

hostname
: Name of host to use in domain name server search

Related Commands

See net-test available, net-test ping, and net-test tcp.

Example

Perform DNS lookup on system1.rtp.ibm.com.

```
Console> net-test dns system1.rtp.raleigh.ibm.com
9.42.77.42
Console>
```

Parent topic: Special purpose commands

net-test ping

Purpose

Test connectivity to a specified host name or IP address.

Syntax

net-test ping *host*

Parameters

host
: Host name or IP address of the remote system

Related Commands

For other diagnostic commands, see net-test available, net-test dns, and net-test tcp.

Example

Test connectivity to a specified host name. The remote system is reachable.

```
Console> net-test ping system1.rtp.raleigh.ibm.com
Ok
Console>
```

Test connectivity to a specified address. The remote system is not reachable.

```
Console> net-test ping 4.4.4.4
ping failed
Console>
```

Parent topic: Special purpose commands

net-test tcp

Purpose

Test ability to open a TCP connection to port at a specified host name or IP address.

If successfully opened, no data is sent across the TCP connection, it is closed.

Syntax

net-test tcp *host port*

Parameters

host
: Host name or IP address of the remote system

port
: TCP port at the remote system

Related Commands

For other diagnostic commands, see net-test available, net-test dns, and net-test ping.

Example

Test ability to reach port 80 at host system1.rtp.raleigh.ibm.com.

```
Console> net-test tcp system1.rtp.raleigh.ibm.com 80
Ok
```

Parent topic: Special purpose commands

net-test traceroute

Purpose

Report path to a specified hostname or IP address.

If successfully opened, no data is sent across the TCP connection, it is closed.

Syntax

net-test traceroute *host*

Parameters

host
: Host name or IP address of the remote system

Related Commands

For other diagnostic commands, see net-test available, net-test dns, net-test ping, and net-test tcp.

Example

Determine route to specified host.

```
Console> net-test traceroute localhost
localhost (127.0.0.1)  0.097 ms
Ok
Console>
```

```
Console> net-test traceroute 127.0.0.1
localhost (127.0.0.1)  0.099 ms
Ok
Console>
```

Test connectivity to specified address. Remote system is not reachable.

```
Console> net-test traceroute host.that.does.not.exist
bad address 'host.that.does.not.exist'
traceroute failed
Console>
Console> net-test traceroute 9.65.47.137
sendto: Network is unreachable
traceroute failed
```

Parent topic: Special purpose commands

nodename get

Purpose

Retrieve the configured name of an appliance.

Syntax

nodename get

Parameters

None.

Related Commands

See nodename set.

Example

Set and then retrieve the appliance name.

```
Console> nodename set appliance1
Console> nodename get
nodename is appliance1
Console>
```

Parent topic: Special purpose commands

nodename set

Purpose

Set the configured name of an appliance.

Syntax

nodename set *name*

Parameters

name
: Name of appliance

Related Commands

See nodename get.

Example

Set and then retrieve the appliance name.

```
Console> nodename set appliance1
Console> nodename get
nodename is appliance1
Console>
```

Parent topic: Special purpose commands

packet-capture clear

Purpose

Remove packet trace capture files for specified network interface.

Syntax

packet-capture clear *interface*

Parameters

interface
: Network interface for which packet trace files need to be cleared

Usage Notes

- After a packet trace is run on an interface, the files must be cleared before another packet capture can be started for the same interface.

Related Commands

See packet-capture list, packet-capture start, and packet-capture stop.

Example

Display packet capture activity and then clear the files collected for interface `eth0`.

```
Console> packet-capture list
interface       status          file(s)
eth1    Finished                eth1pc1
                                   eth1pc2
------------------------------
eth0    Finished                eth0pc
Ok
Console> packet-capture clear eth0
Packet capture dump files cleared for eth0
Ok
Console>
```

Parent topic: Special purpose commands

packet-capture list

Purpose

Show packet-capture activity on the appliance.

Syntax

packet-capture list

Parameters

None.

Related Commands

See packet-capture clear, packet-capture start, and packet-capture stop.

Example

Show packet-capture activity on the appliance. In this example, packet-capture has finished for interface **eth0**. For interface **mgt0** trace is still being actively written to three ring buffer files.

```
Console> packet-capture list
interface       status          file(s)
eth0    Finished                eth0pc
--------------------------------
mgt0    Capturing               mgt0pc1
                                mgt0pc2
                                mgt0pc3
--------------------------------
Ok
Console>
```

Parent topic: Special purpose commands

packet-capture start

Purpose

Activate packet trace capture.

Syntax

packet-capture start *interface filename duration filesize* [*ring-buffer-number*] ["*filter* "]

Parameters

interface
: Network interface name, such as **eth0**

filename
: Output trace file name. If *ring-buffer-number* is also specified, an index number is appended to the filename.

duration
: Packet capture duration, in seconds. A value of **0** indicates the packet capture should continue until the packet-capture stop command is issued.

filesize
: Size of trace file, in megabytes (MB). The value must be from 0 through 100.

ring-buffer-number
: If specified, enables ring buffer function and sets number of ring buffer files. The value must be from 0 through 10.

filter
: Filter to use in selecting captured packets, within double quotation marks (\" \"). An extensive filter capability is supported, including, for example, filtering on packet source, destination, ports, and protocols. Search the internet for pcap-filter for complete details on supported filter syntax.

Usage Notes

- In order to start a packet capture on an interface, the following conditions are required:
  - The interface must be active.
  - No other packet captures can be active for the interface.
  - If tracing is required for an aggregate-interface, start the trace on the aggregate-interface itself. A packet capture on a member link may not contain complete data.
  - Any files from a previous packet capture must be cleared by using the packet-capture clear command.
- Packet capture stops when any of the following condition occurs:
  - The file size limitation is reached, if the ring-buffer function is not used.
  - The packet capture duration ends.
  - The packet-capture stop command is issued.
- Output trace files can be analyzed with commercial off-the-shelf tools that can read data in libpcap format, such as Wireshark.

Related Commands

See packet-capture clear, packet-capture list, and packet-capture stop.

Example

Start a packet capture on interface `eth1` for 60 seconds. Allow the trace files to grow to a

maximum of 10 MB. Write the packet capture to two ring buffer files named `pc11` and `pc12`.

```
Console> packet-capture start eth1 pc1 60 10 2 "icmp"
Packet capture started on eth1
Ok
Console>
```

Parent topic: Special purpose commands

packet-capture stop

Purpose

Stop an in-progress packet capture for a specified network interface.

Syntax

packet-capture stop *interface*

Parameters

interface
: Network interface for which packet capture needs to be stopped

Usage Notes

- Packet capture files remain in place until the packet-capture clear command is issued.
- Use file put to transfer the packet-capture files off the appliance for analysis.

Related Commands

See packet-capture clear, packet-capture list, and packet-capture start.

Example

Display active packet capture and then stop packet-capture activity on interface `eth1`.

```
Console> packet-capture list
interface       status          file(s)
eth1    Capturing               eth1pc1
                                eth1pc2
-------------------------------
eth0    Finished                eth0pc
-------------------------------
Ok
Console> packet-capture stop eth1
Stopping capture on eth1
Ok
Console>
```

Parent topic: Special purpose commands

platform collect-pd

Purpose

Collect a standard set of problem diagnosis data into a single file. This file can be transferred off the appliance for viewing or for sending to IBM support. The resulting output file is the output of a predefined list of command line interface (CLI) commands. The intention of this command is to "script" a number of command line interface (CLI) commands a support person might ask the customer to run as part of problem diagnosis.

Syntax

platform collect-pd [*PDfilename*]

Parameters

PDfilename
: Specifies the name of the text file to which the output of the problem diagnostic commands is written. If not specified, the default value is collect-pd.txt. If the output file exists, it is overwritten.

Usage Notes

- The output might include both clear text that customers can read, and obscured textual data intended only for IBM support.
- The output of this command is also included automatically in the data collected by the platform must-gather command. So if users run platform must-gather, they do not need to separately run platform collect-pd.
- Because the output is all printable text, the contents of the output file might be printed inside the command line interface (CLI) session by using the show log command. This is helpful if the appliance loses all network connectivity.

Related Commands

See platform must-gather, file put, and show log.

Example

Collects diagnostic command data, sending the output to the default output file.

```
Console> platform collect-pd
Console> file list
collect-pd.txt 79336 bytes created Feb 6, 2012 9:32:17 PM
Console>
```

Parent topic: Special purpose commands

platform log-level get

This command is deprecated. Instead, use show logging-settings.
Parent topic: Special purpose commands

platform log-level set

This command is deprecated. Instead, use edit logging-settings.
Parent topic: Special purpose commands

platform service disable ssh

Purpose

- Disable an ssh platform service if it affects your security policies.

Syntax

platform service ssh ... platform service ssh disable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service enable ssh, platform service enable telnet, or platform service disable telnet.

Example

```
Console> help platform service
CWZBR02449I: platform service ssh ...
CWZBR02901I: platform service ssh disable
```

Parent topic: Special purpose commands

platform service disable telnet

Purpose

- Disable a telnet platform service security protocol.

Syntax

platform service telnet ... platform service telnet disable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service enable telnet, platform service enable ssh, or platform service disable ssh.

Example

```
Console> help platform service
CWZBR02449I: platform service telnet ...
CWZBR02901I: platform service telnet disable
```

Parent topic: Special purpose commands

platform service enable ssh

Purpose

- Enable an ssh platform service security protocol.

Syntax

platform service ssh ... platform service ssh enable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service disable ssh, platform service enable telnet, or platform service disable telnet.

Example

```
Console> help platform service
CWZBR02449I: platform service ssh ...
CWZBR02901I: platform service ssh enable
```

Parent topic: Special purpose commands

platform service enable telnet

Purpose

- Enable a telnet platform service security protocol.

Syntax

platform service telnet ... platform service telnet enable

Parameters

None.

Usage Notes

None.

Related Commands

See platform service disable telnet, platform service enable ssh, or platform service disable ssh.

Example

```
Console> help platform service
CWZBR02449I: platform service telnet ...
CWZBR02901I: platform service telnet enable
```

Parent topic: Special purpose commands

platform must-gather

Purpose

Collect the must-gather diagnostic data required for reporting a problem to IBM support.

Syntax

platform must-gather *filename* [*PDfilename*]

Parameters

filename
: Output file containing all diagnostic data, in a compressed (.tgz) format

pdfilename
: Name of the text file to which the output of the problem diagnostic commands will be written. This file contains the same output collected by the `platform collect-pd` command. If not specified, the default value is `collect-pd.txt`. If the output file exists, the file is overwritten.

Usage Notes

- The output might include both clear text and obscured data intended to be read only by IBM support.

Related Commands

See platform collect-pd.

Example

Collect diagnostic data to a file called `problem-data.tgz`.

```
Console> platform must-gather problem-data.tgz
Console> file list
problem-data.tgz 3449276 bytes created Mar 27, 2012 11:15:48 AM
collect-pd.txt 572878 bytes created Mar 27, 2012 11:15:48 AM
Console>
```

Parent topic: Special purpose commands

raid delete

Purpose

Delete the configuration on the RAID card controller and also clean up volume and partition information to set the physical disk back to factory state.

Syntax

raid delete

Parameters

None.

Usage Notes

- The data on the disk will be lost.
- The configuration on the RAID card controller will be erased and will be automatically reconfigured at the next boot.
- The partition will be eliminated and the partition information will be erased on the disk.
- The state of all physical disks will be reset.
- Only volumes managed by the RAID card controller will be deleted.
- Unmanaged volumes will not be deleted.

Example

Delete configuration on the RAID controller.

```
Console> raid delete
Delete volume
Delete volume successfully
Console> status volume
No volume
Console> status raid physical
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 0
    Logical drive ID: 0
    Logical drive name: raid0
    State: unconfigured good drive
...
```

Parent topic: Special purpose commands

raid re-use

Purpose

Make a physical disk previously used in another RAID card controller available for use in the appliance. When a previously used disk is inserted into the appliance, it might be recognized by the RAID controller as a foreign disk. This command will make the disk usable by the RAID controller and automatically start the rebuilding process for the disk into the RAID array.

Syntax

raid re-use

Parameters

None.

Usage Notes

- The old data on the physical disk will be lost.
- This command will clear the foreign state of a used physical disk and also trigger the rebuilding process. After the building process completes, the physical disk will transition online.
- This command works for only one physical disk. If more than one foreign physical disk is inserted at a time, the command will not be able to make them into a RAID configuration.

Example

Clean used physical disk foreign state and start rebuilding.

```
Console> raid re-use
Bring physical drive back online after hot swap
Done
Console> status raid physical
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: rebuild
    Progress percent: 3
...
```

Parent topic: Special purpose commands

raid foreign-config import

Purpose

Importing a foreign configuration is supported on 9005. A foreign configuration is a RAID configuration that already exists on a replacement set of drives. Foreign configurations on drives can be cleared or imported into the appliances on which they are installed. By default, foreign configurations are cleared if they are found on a 9005 machine. Users may enable the importing of foreign configurations on their needs via CLI command or C++/Java API.

Syntax

raid foreign-config import

Parameters

None.

Usage Notes

To import the foreign configurations on the replacement disks, take the following steps in order:

1. Execute the command raid foreign-config import to enable the importing of foreign configurations.
2. Shutdown the appliance.
3. Replace its drives with the ones on which a foreign configuration already exists.

After powering on the appliance, execute status volume to see if the raid volume is online. And check if the foreign configuration is successfully imported. By default, the importing of the foreign configurations is disabled. When it is disabled, any foreign configurations found on the disks will be cleared. Data on the disks will be lost too when a RAID configuration is cleared.

Related Commands

See raid foreign-config clear.

Example

Import a foreign configuration on the RAID controller.

```
Console> raid foreign-config import
CWZBR02804I: Import of foreign configuration is enabled. Foreign configurations will be imported a
t reboot.
```

Parent topic: Special purpose commands

raid foreign-config clear

Purpose

Import of foreign configuration is disabled. Foreign configurations will be cleared at reboot. Importing a foreign configuration is supported on 9005. A foreign configuration is a RAID configuration that already exists on a replacement set of drives. Foreign configurations on drives can be cleared or imported into the appliances on which they are installed. By default, foreign configurations are cleared if they are found on a 9005 machine.

Syntax

raid foreign-config clear

Parameters

None.

Related Commands

See raid foreign-config import.

Example

Import a foreign configuration on the RAID controller is disabled.

```
Console> raid foreign-config clear
CWZBR02805I: Import of foreign configuration is disabled. Foreign configurations will be cleared a
t reboot.
```

Parent topic: Special purpose commands

request force-suspend

Purpose

Request an abrupt suspension of grid activities while maintaining a network connection to the grid. If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is still available.

Syntax

request force-suspend

Options

None.

Parameters

None.

Usage Notes

None.

Related Commands

See request resume and request suspend.

Example

```
Console> request force-suspend
Force Stopped all XC-10 processes
Forced recycle of appliance
[Mon Feb 25 2013 09:46:59]  request force-suspend completed
Console> start-progresss
CWZBR02115E: Unknown command "start-progress"
Console> start-progress
DNS 10.42.229.86 10.42.229.86
DNS 10.1.1.105 10.1.1.105
DNS 9.42.94.15 9.42.94.15
SUSPENDED
```

Parent topic: Special purpose commands

request resume

Purpose

If you have previously requested a suspension of grid activities either through a request suspend or a request force suspend command, then this command restores grid activities.

Syntax

request resume

Options

None.

Parameters

None.

Usage Notes

This command takes some time to completely restore activities. If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is still available.

Related Commands

See request suspend and request force-suspend.

Example

```
Console> request resume
[Mon Feb 25 2013 09:48:09]  request resume completed
Console> start-progress
DNS 10.42.229.86 10.42.229.86
DNS 10.1.1.105 10.1.1.105
DNS 9.42.94.15 9.42.94.15
Volume 1 mounted
Volume 2 mounted
STARTING (4% complete)
```

Parent topic: Special purpose commands

request suspend

Purpose

Request a graceful suspension of grid activities while maintaining a network connection to the grid. If you are connected to the command line interface (CLI) through a network connection (i.e., ssh or telnet), your network connection is still available.

Syntax

request suspend

Options

None.

Parameters

None.

Usage Notes

None.

Related Commands

See request resume and request force-suspend.

Example

```
Console> request suspend
Stopped all XC-10 processes
Recycle of appliance
[Mon Feb 25 2013 09:46:59]  request suspend completed
Console> start-progresss
CWZBR02115E: Unknown command "start-progress"
Console> start-progress
DNS 10.42.229.86 10.42.229.86
DNS 10.1.1.105 10.1.1.105
DNS 9.42.94.15 9.42.94.15
SUSPENDED
```

Parent topic: Special purpose commands

set-dns-search

Purpose

Configure domain name server search domains.

Syntax

set-dns-search [*domain* [... *domain*]]

Parameters

domain
: Domain name server search domains.

Usage Notes

- Invoking the set-dns-search command without specifying a domain clears the list.
- Each command invocation replaces the previously configured domains.

Related Commands

See get-dns-search, get-ntp-servers, and set-dns-servers.

Example

Configure and then display domain name server search domains.

```
Console> set-dns-search rtp.raleigh.ibm.com raleigh.ibm.com ibm.com
Ok
Console> get-dns-search
DNS search domains:
        rtp.raleigh.ibm.com
        raleigh.ibm.com
        ibm.com
Console>
```

Parent topic: Special purpose commands

set-dns-servers

Purpose

Configure IP addresses of one or more domain name servers.

Syntax

set-dns-servers [*server* [... *server*]]

Parameters

server
: IP address of the domain name server

Usage Notes

- Starting the set-dns-servers command without specifying a server clears the list.
- Each command invocation replaces the previously configured servers.

Related Commands

See get-dns-servers, get-dns-search, and set-dns-search.

Example

Configure and then display domain name servers.

```
Console> set-dns-servers 9.42.106.2  2002:92a:8f7a:106:9:42:106:42
Ok
Console> get-dns-servers
Domain (DNS) servers:
        9.42.106.2
        2002:92a:8f7a:106:9:42:106:42
Console>
```

Parent topic: Special purpose commands

set-ntp-servers

Purpose

Configure network time protocol servers.

Syntax

set-ntp-servers [*server* [... *server*]]

Parameters

server
: IP address or host name of the network time protocol server

Usage Notes

- Starting the set-ntp-servers command without specifying a server clears the list.
- Each command invocation replaces the previously configured servers.

Related Commands

See get-ntp-servers and timezone set.

Example

Configure and then display NTP servers.

```
Console> set-ntp-servers serverA.yourco.com 10.10.106.44
Ok
Console> get-ntp-servers
NTP servers:    serverA.yourco.com
        10.10.106.44
Console>
```

Parent topic: Special purpose commands

show commandref

Purpose

Display a reference list of all commands supported in the command line interface (CLI).

Syntax

show commandref

Parameters

None.

Usage Notes

- Output from this command is intended to be parsed by a program.
- Output might be lengthy, so use of the command from the serial console is not suggested.

Example

List commands.

```
Console> show commandref
<reference>

<commands>
  <entry>
    <name>alias</name>
    <brief>alias [name [value]]/brief>
    <detailed/>
  </entry>
......
```

Parent topic: Special purpose commands

show components

Purpose

Display component firmware versions on the appliance.

Syntax

show components

Parameters

None.

Usage Notes

None.

Example

Display firmware versions of various components.

```
Console> show components
BMC Firmware Version: 2.53
ServeRaid Firmware Version: 12.12.0-0039
```

Parent topic: Special purpose commands

show locales

Purpose

Display a list of language, country and charset values usable for the locale set command. The valid combinations of language and country are displayed, along with the default charset for that combination. Then the entire list of supported charsets is displayed.

Syntax

show locales

Parameters

None.

Usage Notes

Some languages have multiple dialects, for example Chinese (zh) has simplified (Hans) and traditional (Hant) dialects. To specify traditional Chinese, use language zh_Hant. Be sure to specify a charset that is supported by the client software that you will be using to communicate to the appliance.

Related Commands

None.

Example

List the locale values usable for the locale set command.

```
 Console> show locales
   CWZBR02866I: af
   CWZBR02866I: af NA UTF-8
   CWZBR02866I: af ZA UTF-8
   CWZBR02866I: agq
   CWZBR02866I: agq CM UTF-8
   ...
   CWZBR02866I: zh TW BIG5
   CWZBR02866I: zu
   CWZBR02866I: zu ZA UTF-8
   CWZBR02867I: supported charsets: Big5 EUC-JP EUC-KR GB18030 GB2312 IBM00858 IBM850 IBM857 IBM86
0
   IBM861 IBM862 IBM863 IBM864 IBM865 IBM866 IBM868 IBM869 ISO-2022-JP ISO-2022-JP-1 ISO-2022-JP-2
   ISO-2022-KR ISO-8859-1 ISO-8859-10 ISO-8859-13 ISO-8859-15 ISO-8859-2 ISO-8859-3 ISO-8859-4
   ISO-8859-5 ISO-8859-6 ISO-8859-7 ISO-8859-8 ISO-8859-9 KOI8-R KSC_5601 Shift_JIS US-ASCII UTF-1
6
   UTF-16BE UTF-16LE UTF-32 UTF-7 UTF-8 cp1363 cp851 macintosh x-mac-centraleurroman x-mac-cyrilli
c
   x-mac-greek x-mac-turkish
```

Parent topic: Special purpose commands

show log

Purpose

Display a list of logfiles or the contents of a log file.

Syntax

- Display the list of log files: show log list
- Display the contents of a log file: show log *filename*

Parameters

filename
: Name of file to be listed to console

Usage Notes

- For reporting problems to IBM Support, use the platform must-gather command.
- In the event that network problems make it impossible to retrieve the must-gather output, this command can be used to list the logs to the serial console. However, be aware that the serial port is slow and the log files might be very large. Listing them this way might take a long time.

Example

List the displayable log files and then list the contents of the default-log file.

```
Console> show log list
default-log
default-trace
Console> show log default-log
2012-02-17T19:25:43+00:00 CWZBR00028 [info][firmwaremgr(3929)]: The Firmware Manager has started.
2012-02-17T19:25:43+00:00 CWZBR00031 [debug][firmwaremgr(3929)]: The Firmware Manager was created.
...
```

Parent topic: Special purpose commands

show timezones

Purpose

Display a list of time zones usable for the `timezone set` command.

Syntax

show timezones

Parameters

None.

Example

List the timezone values usable for the timezone set command.

```
Console> show timezones
EST5EDT
PST8PDT
...
```

Parent topic: Special purpose commands

show version

Purpose

Display the firmware version active on the appliance.

Syntax

show version

Parameters

None.

Example

Display the active firmware version.

```
Console> show version
Installation date: Mar 9, 2012 8:06:27 PM
Platform version: 4.9.4.0
Platform build ID: build12-20120309-1307
Platform build date: 2012-03-09 18:09:44+00:00
Machine type/model: 923572X
Serial number: 68A0553
Firmware type: Development
Console>
```

Parent topic: Special purpose commands

sshkey fingerprint

Purpose

Display the server keys used by the SSH server of the appliance.

Syntax

sshkey fingerprint

Parameters

None.

Example

Display SSH keys of the server.

```
Console> sshkey fingerprint
SSH RSA Key fingerprint 10:c6:c6:5c:44:72:d1:e9:84:15:df:65:47:96:1a:c1
SSH DSA Key fingerprint cd:a9:8e:a8:0b:d0:45:ce:ac:9e:67:92:06:cf:63:1b
Console>
```

Parent topic: Special purpose commands

sshkey reset

Purpose

Delete and regenerate the server keys used by the appliance's SSH server.

Syntax

sshkey reset

Parameters

None.

Related Commands

None.

Example

Reset the server's SSH keys.

```
Console> sshkey fingerprint
CWZBR02154I: SSH RSA Key fingerprint 10:c6:c6:5c:44:72:d1:e9:84:15:df:65:47:96:1a:c1
CWZBR02154I: SSH DSA Key fingerprint cd:a9:8e:a8:0b:d0:45:ce:ac:9e:67:92:06:cf:63:1b
Console> sshkey reset
CWZBR02196I: Ok
Console> sshkey fingerprint
CWZBR02154I: SSH RSA Key fingerprint 7c:43:c7:36:2e:4a:ed:2d:a9:af:62:cc:44:f0:a5:7b
CWZBR02154I: SSH DSA Key fingerprint c3:fb:08:aa:c8:81:81:57:f1:ff:9b:c3:70:df:4a:bd
Console>
```

Parent topic: Special purpose commands

sshkey verify

Purpose

Determine whether the server keys used by the appliance's SSH server are strong or weak.

Syntax

sshkey verify

Parameters

None.

Related Commands

None.

Example

Verify the strength of the server's SSH keys and reset them if they are not optimally strong.

```
Console> sshkey verify
CWZBR02797E: SSH keys are not optimally strong
Console> sshkey reset
CWZBR02196I: Ok
Console> sshkey verify
CWZBR02798I: Host SSH keys are strong
Console>
```

Parent topic: Special purpose commands

start-progress

Purpose

You can issue this command when the startup is in progress. The command displays the percentage of the startup process that has completed.

Syntax

start-progress
None.

Parameters

None.

Related Commands

None.

Example

```
Console> start-progress
DNS 9.42.139.134 9.42.139.134
DNS 9.42.139.139 9.42.139.139
DNS 9.42.139.152 xsa23e2.rtp.raleigh.ibm.com
DNS 9.42.139.142 localhost
CWXSA0014I: Volume 1 mounted
CWXSA0014I: Volume 2 mounted
CWXSA0006I: Catalog server started.
CWXSA0018I: Grid configuration service started.
CWXSA0017I: Container server 02 started.
CWXSA0017I: Container server 07 started.
CWXSA0017I: Container server 04 started.
CWXSA0017I: Container server 06 started.
CWXSA0017I: Container server 08 started.
CWXSA0017I: Container server 05 started.
CWXSA0017I: Container server 01 started.
CWXSA0017I: Container server 03 started.
CWXSA0002I: Grid administrative service started.
CWXSA0004I: Administrative console started.
STARTED
```

Parent topic: Special purpose commands

status battery

Purpose

Display expected battery expiration.

Syntax

status battery

Parameters

None.

Usage Notes

- The battery provides backup power for the hardware configuration and RAID disk writeable cache while the appliance is powered off and unplugged from an AC power source. Batteries are valid for a specified duration. This command displays the start and end of that duration.

Related Commands

See also status battery.

Example

Display expected battery expiration.

```
Console> status battery
Battery installed: Feb 17, 2012 7:25:45 PM
Battery expires: Feb 16, 2014 7:25:45 PM
Console>
```

Parent topic: Special purpose commands

status cpu-usage

This command is deprecated. Use status cpu-utilization instead.
Parent topic: Special purpose commands

status cpu-utilization

Purpose

Display CPU utilization over the past 10 seconds, 1 minute, 10 minutes, 1 hour, and 1 day intervals.

Syntax

status cpu-utilization

Parameters

None.

Usage Notes

None.

Example

Display CPU utilization.

```
Console> status cpu-utilization
CPU utilization over time:
0% over 10 seconds
49% over 1 minute
25% over 10 minutes
14% over 1 hour
6% over 1 day
Console>
```

Parent topic: Special purpose commands

status dynamic-tunnels

Purpose

Display details of currently active dynamic IPsec tunnels.

Syntax

status dynamic-tunnels *[tunnelID]*

Parameters

tunnelID
: Optional identifier of a specific dynamic tunnel to display in the form *Ynnn*. If a tunnel identifier is not specified, all of the existing dynamic IPsec tunnels are displayed.

Usage Notes

See "Network Security Configuration" for details about the fields returned by this status provider.

Example

Display the status of dynamic tunnel Y2.

```
Console> status dynamic-tunnels y2
Dynamic tunnel name: Y2
        AssociatedFiltCode: n/a
        AssociatedFiltCodeRange: n/a
        AssociatedFiltDestPort: n/a
        AssociatedFiltDestPortRange: n/a
        AssociatedFiltProtocol: ALL(0)
        AssociatedFiltSrcPort: n/a
        AssociatedFiltSrcPortRange: n/a
        AssociatedFiltType: n/a
        AssociatedFiltTypeRange: n/a
        AuthAlgorithm: HMAC-SHA1
        AuthInboundSpi: 4258054942 (0xFDCCC31E)
        AuthOutboundSpi: 796243132  (0x2F75B4BC)
        CurrentTime: 2012/06/01 15:27:14
        DiffieHellmanGroup: 14
        EncryptInboundSpi: 4258054942 (0xFDCCC31E)
        EncryptOutboundSpi: 796243132  (0x2F75B4BC)
        Generation: 1
        HowActivated: OnDemand
        HowToAuth: ESP
        HowToEncap: Transport
        HowToEncrypt: 3DES-CBC
        IKEVersion: 2.0
        InboundBytes: 192
        InboundPackets: 3
        IpFilterRule: tcpcs-ipsec-ifr
        KeyLength: n/a
        Lifesize: 0K
        LifesizeRefresh: 0K
        LifetimeExpires: 2012/06/01 19:17:54
        LifetimeRefresh: 2012/06/01 19:13:05
        LocalAddressBase: 9.42.90.6
        LocalAddressPrefix: n/a
        LocalAddressRange: n/a
```

```
LocalDynVpnRule: n/a
LocalEndPoint: 9.42.90.6
OutboundBytes: 192
OutboundPackets: 3
PFS: Yes
ParentIKETunnelID: K1
PendingNewActivation: n/a
RemoteAddressBase: 9.42.105.155
RemoteAddressPrefix: n/a
RemoteAddressRange: n/a
RemoteEndPoint: 9.42.105.155
RmtIpSpecExIDPayload: n/a
State: DONE
TunnelID: Y2
VpnActionName: tcpcs-idva
Console>
```

Parent topic: Special purpose commands

status fan

Purpose

Display fan speeds and status.

Syntax

status fan

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms, or blade-based platforms.

Example

Display fan status.

```
Console> status fan
Fan Speed Tray 1 Fan 1: 4800 (ok)
Fan Speed Tray 1 Fan 2: 5840 (ok)
Fan Speed Tray 1 Fan 3: 0.000000 (alert)
Fan Speed Tray 1 Fan 4: 6960 (ok)
Fan Speed Tray 2 Fan 1: 4720 (ok)
Fan Speed Tray 2 Fan 2: 5520 (ok)
Fan Speed Tray 2 Fan 3: 4720 (ok)
Fan Speed Tray 2 Fan 4: 6080 (ok)
Hard Disk Tray Fan 1: 4720 (ok)
Hard Disk Tray Fan 2: 6000 (ok)
Console>
```

Parent topic: Special purpose commands

status flash

Purpose

Display size and available space on the flash drive.

Syntax

status flash

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms.

Example

Display flash memory usage.

```
Console> status flash
Flash drive status:
        Total size = 3908 MB, free size = 1894 MB
Console>
```

Parent topic: Special purpose commands

status intrusion

Purpose

Display integrity status of physical appliance case.

Syntax

status intrusion

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms, or blade-based platforms.

Related Commands

See device intrusion clear.

Example

Display case integrity status.

```
Console> status intrusion
Case has not been opened and is secure
Console>
```

Parent topic: Special purpose commands

status memory

Purpose

Display RAM memory usage.

Syntax

status memory

Parameters

None.

Example

Display memory usage.

```
Console> status memory
Memory size 8193636K, free 7696564K
Console>
```

Parent topic: Special purpose commands

status nvdimm

Purpose

Display nvDIMM state.

Syntax

status nvdimm

Parameters

None.

Example

Display nvDIMM status.

```
Console> status nvdimm
  CWZBR02911I: All nvDIMMs are ready to go
  Console>
```

Parent topic: Special purpose commands

status power-supply

Purpose

Display status of power supply units.

Syntax

status power-supply

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms, or blade-based platforms.

Example

Display status of power supply units.

```
Console> status power-supply
Power Supply 1 Output Failure: true (alert)
Power Supply 2 Output Failure: false (ok)
Console>
```

Parent topic: Special purpose commands

status raid all

Purpose

Display details of physical disks and logical disks that are controlled by the RAID card controller.

Syntax

status raid all

Parameters

None.

Usage Notes

- Logical disk information here is not supported yet, so only physical disk detail is displayed.

Related Commands

See status volume and status raid physical.

Example

Show all physical disks and logical disks controlled by the RAID card controller.

```
Console> status raid all
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
    Interface type: SAS
    Interface speed: 6.0Gb/s
    SAS address: 5000c500286968d50000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE2CE1F
Position: HDD0
    Controller ID: 1
    Device ID: 9
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
    Interface type: SAS
```

```
    Interface speed: 6.0Gb/s
    SAS address: 5000c500286970090000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE28HV9
Console>
```

Parent topic: Special purpose commands

status raid battery

Purpose

Display the RAID BBU information.

Syntax

status raid battery

Parameters

None.

Usage notes

The BBU information is available only on appliances with MegaRAID controller, but not with MPT controller.

Example

Display the RAID BBU information.

```
Console> status raid battery
 CWZBR02808I: BBU status for Adapter '0':
 CWZBR02809I:   Battery Type                                : iBBU
 CWZBR02810I:   Voltage                                     : 4055 mV
 CWZBR02811I:   Current                                     : 0 mA
 CWZBR02812I:   Temperature                                 : 29 C
 CWZBR02816I: BBU Firmware Status:
 CWZBR02819I:   Charging Status                             : None
 CWZBR02820I:   Voltage                                     : OK
 CWZBR02821I:   Temperature                                 : OK
 CWZBR02822I:   Learn Cycle Requested                       : No
 CWZBR02823I:   Learn Cycle Active                          : No
 CWZBR02824I:   Learn Cycle Status                          : OK
 CWZBR02825I:   Learn Cycle Timeout                         : No
 CWZBR02826I:   I2c Errors Detected                         : No
 CWZBR02827I:   Battery Pack Missing                        : No
 CWZBR02828I:   Battery Replacement required                : No
 CWZBR02829I:   Remaining Capacity Low                      : No
 CWZBR02830I:   Periodic Learn Required                     : No
 CWZBR02831I:   Transparent Learn                           : No
 CWZBR02832I:   No space to cache offload                   : No
 CWZBR02833I:   Pack is about to fail. Should be replaced   : No
 CWZBR02834I:   Cache Offload premium feature required      : No
 CWZBR02835I:   Module microcode update required            : No
 CWZBR02847I: BBU Capacity Info:
 CWZBR02848I:   Relative State of Charge                    : 98 %
 CWZBR02849I:   Absolute State of Charge                    : 96 %
 CWZBR02850I:   Remaining Capacity                          : 1163 mAh
 CWZBR02851I:   Full Charge Capacity                        : 1187 mAh
 CWZBR02853I:   Run time to empty (min)                     : Battery is not being discharged
 CWZBR02855I:   Average time to empty (min)                 : Battery is not being discharged
 CWZBR02857I:   Average Time to full (min)                  : Battery is not being charged
 CWZBR02858I:   Cycle Count                                 : 21
 CWZBR02859I:   Max Error                                   : 6 %
 CWZBR02860I:   Remaining Capacity Alarm                    : 120 mAh
 CWZBR02861I:   Remaining Time Alarm                        : 10 Min
 CWZBR02836I: BBU Design Info:
```

```
CWZBR02837I:   Device Name                          : 3150301
CWZBR02838I:   Serial Number                        : 4010
CWZBR02839I:   Manufacturer Name                    : LS1121001A
CWZBR02840I:   Date of Manufacture                  : 07-03-2010
CWZBR02841I:   Design Capacity                      : 1215 mAh
CWZBR02843I:   Design Voltage                       : 3700 mV
CWZBR02844I:   Transparent Learn                    : 0
CWZBR02845I:   Specification Info                   : 33
CWZBR02846I:   Pack Stat Configuration              : 0x6490
Console>
```

Parent topic: Special purpose commands

status raid physical

Purpose

Display details of physical disks that are controlled by the RAID card controller.

Syntax

status raid physical

Parameters

None.

Usage Notes

- Detailed information displayed will be different based on different RAID card controllers.
- The *Location* or *Position* information corresponds with the physical location of the disks on the appliance. On the hard drive tray or front panel of the appliance, text is provided with this location information.

Related Commands

See status volume and status raid all.

Example

Show all physical disks controlled by the RAID card controller.

```
Console> status raid physical
Raid physical disk status:
Position: HDD1
    Controller ID: 1
    Device ID: 8
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
    Interface type: SAS
    Interface speed: 6.0Gb/s
    SAS address: 5000c500286968d500000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE2CE1F
Position: HDD0
    Controller ID: 1
    Device ID: 9
    Array ID: 1
    Logical drive ID: 1
    Logical drive name: raid0
    State: online
    Progress percent: 0
    Raw size: 286102
    Normalized size: 285148
```

```
    Interface type: SAS
    Interface speed: 6.0Gb/s
    SAS address: 5000c5002869700900000000000000000
    Vendor ID: SEAGATE
    Product ID: ST9300603SS
    Raid Firmware Version: 12.12.0-0065
    Revision: 0006
    Vendor specific information: 3SE28HV9
Console>
```

Parent topic: Special purpose commands

status voltage

Purpose

Display voltage readings and status.

Syntax

status voltage

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- Not applicable to virtual platforms.

Example

Display voltage readings.

```
Console> status voltage
Voltage +12: 11.8 (ok)
Voltage +5 Standby: 4.90 (ok)
Voltage +5: 4.87 (ok)
Voltage +3.3: 3.33 (ok)
Voltage +1.8: 1.79 (ok)
Voltage +1.5: 1.49 (ok)
Voltage CPU1 Core: 1.07 (ok)
Voltage CPU2 Core: 1.08 (ok)
Voltage Bus Termination: 1.08 (ok)
Voltage Battery: 3.14 (ok)
Console>
```

Parent topic: Special purpose commands

status volume

Purpose

Display all volume and partition information about the system.

Syntax

status volume

Parameters

None.

Usage Notes

- This command shows both managed and unmanaged types of volumes and partitions.
- Unmanaged partitions does not show the encryption information.
- Unmanaged partitions is shown as offline if the mount point does not exist.

Related Commands

See status raid physical and status raid all.

Example

Show all volume details.

```
Console> status volume
Storage Volume Status:
Volume 'raid0' is [Online] : size = 285148 MB, num partitions = 2
    Partition 1 'raid0-1' is [Online], encryption is [None]
        Total size = 91 MB,   free size = 86 MB
    Partition 2 'raid0-2' is [Online], encryption is [None]
        Total size = 184 MB,   free size = 179 MB
Console>
```

Parent topic: Special purpose commands

status temperature

Purpose

Display temperature status of physical hardware components.

Syntax

status temperature

Parameters

None.

Usage Notes

- Display varies based on the underlying hardware platform.
- This command is not applicable to virtual platforms.

Example

Display temperature status.

```
Console> status temperature
Power Supply 1 Over-Temperature: false (ok)
Power Supply 1 High Temperature: false (ok)
Power Supply 2 Over-Temperature: false (ok)
Power Supply 2 High Temperature: false (ok)
Temperature System 1: 30.0 (ok)
Temperature System 2: 25.0 (ok)
Temperature CPU1: 27.0 (ok)
Temperature CPU2: 30.0 (ok)
Temperature Memory 00: 45.0 (ok)
Temperature Memory 10: 41.0 (ok)
Temperature Memory 20: 39.0 (ok)
Temperature Memory 30: 39.0 (ok)
Temperature Ethernet Ambient: 29 (ok)
Temperature Ethernet MAC: 50 (ok)
Console>
```

Parent topic: Special purpose commands

status uptime

Purpose

Display the length of time since the appliance was booted.

Syntax

status uptime

Parameters

None.

Example

Display length of time appliance has been up.

```
Console> status uptime
13 days 17 hours 13 minutes 15 seconds
Console>
```

Parent topic: Special purpose commands

timezone get

Purpose

Display the configured timezone.

Syntax

timezone get

Parameters

None.

Usage Notes

- None.

Related Commands

See timezone set, datetime get, and datetime set.

Example

Display the configured time zone.

```
Console> timezone get
Timezone is EST
Console>
```

Parent topic: Special purpose commands

timezone set

Purpose

Set the timezone for the local time.

Syntax

timezone set *zone*

Parameters

zone
: Timezone value

Usage Notes

- None.

Related Commands

See timezone get, datetime get, and datetime set.

Example

Set the timezone to Eastern Standard Time (EST) and confirm the setting.

```
Console> timezone set EST
Console> timezone get
Timezone is EST
Console>
```

Parent topic: Special purpose commands

unalias

Purpose

Remove a configured command line interface (CLI) command alias.

Syntax

unalias *name*

Parameters

name
: Alternate name for command line interface (CLI) command that is to be removed

Related Commands

See alias.

Example

Show configured aliases and then remove the alias called `test-connectivity`.

```
Console> alias
alias test-connectivity "net-test ping 9.42.106.2"
Console> unalias test-connectivity
Console> alias
No aliases defined
Console>
```

Parent topic: Special purpose commands

usage

Purpose

Display details about how to use the config command.

Syntax

```
config usage
```

Parameters

None

Usage notes

None

Example

Display details about the config command.

```
config usage
```

Parent topic: Special purpose commands

user add

Purpose

Define a new user to the appliance.

Syntax

user add *username* [*defaultpass*]

Parameters

username
: User name to be added. User names must consist of alphanumeric characters or the special characters: dot(.), dash(-), underscore(_) or plus(+).

defaultpass
: Default password to assign to new user name

Usage Notes

- If the password is not provided on the initial command invocation, the command line interface (CLI) prompts for the password and a confirmation of the password.

Related Commands

See user list, user delete, user password.

Example

Add users `fred` and `barney`.

```
Console> user add fred
enter default password:**********
confirm password:**********
Ok
Console> user add barney rubble
Ok
Console> user list
User names:
     admin
     barney
     fred
Console>
```

Parent topic: Special purpose commands

user delete

Purpose

Delete a user from the appliance.

Syntax

user delete *username*

Parameters

username
: User name to be deleted

Related Commands

See also user add and user list.

Example

Delete user `fred`.

```
Console> user list
User names:
        admin
        fred

Console> user delete fred
Ok
Console> user list
User names:
        admin
Console>
```

Parent topic: Special purpose commands

user known-hosts list

Purpose

List the known hosts for the current user. When the user issues command file get|put scp://user@host/dir, the public key of the host is saved in a local file. Use this command to list the hosts that are saved in that file.

Syntax

user known-hosts list

Parameters

None.

Related Commands

Example

List the known hosts for the currently logged-in user.

```
Console> user known-hosts list
  aaa.example.com,192.168.10.200
  bbb.example.com,192.168.11.201
  Console>
```

Parent topic: Special purpose commands

user known-hosts delete

Purpose

Delete the stored public key for the named host from the known hosts file for the current user. When the user issues command file get|put scp://user@host/dir, the public key of the host is saved in a local file, and used to verify the host's identity on subsequent file get|put scp:// commands. If the host changes its key, that verification can fail. Use this command to delete the public key stored for a known host, so the host's new key can be learned.

Syntax

user known-hosts delete hostname

Parameters

hostname
Host name or IP address of the known hosts entry to be deleted.

Related Commands

See user known-hosts list

Example

Delete the public key for known host aaa.example.com from the currently logged-in user's known hosts list.

```
<p>Console&gt; user known-hosts delete aaa.example.com
  CWZBR02196I: Ok
  Console&gt;</p>
```

Parent topic: Special purpose commands

user list

Purpose

List users defined on the appliance.

Syntax

user list

Parameters

None.

Related Commands

See user add and user delete.

Example

List currently defined users.

```
Console> user list
User names:
        admin
        fred
Console>
```

Parent topic: Special purpose commands

user password

Purpose

Change the password for the currently active user.

Syntax

user password *oldpass newpass*

Parameters

oldpass
: Old (currently active) password

newpass
: New password

Usage Notes

- If the old or new password is not provided on the initial command invocation, the command line interface (CLI) prompts for the appropriate passwords and a confirmation of the new password.

Related Commands

See user add.

Example

Change password three times.

```
Console> user password aaaa bbbb
Changing password...
Ok
Console> user password bbbb
new password:****
confirm password:****
Changing password...
Ok
Console> user password
current password:****
new password:*****
confirm password:*****
Changing password...
Ok
Console>
```

Parent topic: Special purpose commands

user sshkey add

Purpose

Retrieve an SSH public key file from the specified URL for use by the current user.

Syntax

user sshkey add *url*

Parameters

url
: URL of public key file

Usage Notes

- Once the key file is stored, an ssh client with the corresponding private key can ssh in as that user without specifying a password.
- Only one public key is supported per appliance user. Subsequent user sshkey add commands replace the public key file for that appliance user.

Related Commands

See user sshkey delete.

Example

Retrieve a public key for the currently logged-in user.

```
Console> user sshkey add scp://user1@system1.rtp.raleigh.ibm.com:~/mypublickey
user1@system1.rtp.raleigh.ibm.com's password:
mypublickey                                   100%  411     0.4KB/s   00:00
Wrote 411 bytes to local storage
Adding ssh public key file for user wilma
Ok
Console>
```

Parent topic: Special purpose commands

user sshkey delete

Purpose

Delete the stored SSH public key file for the current user.

Syntax

user sshkey delete

Parameters

None.

Related Commands

See user sshkey add.

Example

Delete the public key for the currently logged-in user.

```
Console> user sshkey delete
Deleting ssh public key file for user wilma
Ok
Console>
```

Parent topic: Special purpose commands

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

Packages	
com.ibm.websphere.objectgrid	These are the main application APIs for the ObjectGrid.
com.ibm.websphere.objectgrid.client	This package contains the primary interfaces and classes for setting behaviors for ObjectGrid clients for this process.
com.ibm.websphere.objectgrid.config	This package contains the interfaces and a factory class for creating ObjectGrid configuration objects programatically.
com.ibm.websphere.objectgrid.continuousquery	
com.ibm.websphere.objectgrid.continuousquery.exception	
com.ibm.websphere.objectgrid.continuousquery.filter	These are the Filter implementations which are used to build continuous query definitions.
com.ibm.websphere.objectgrid.management	This package contains the interfaces for all ObjectGrid MBeans.
com.ibm.websphere.objectgrid.plugins	These are the interfaces for adding plugins to the Grid core framework.
com.ibm.websphere.objectgrid.plugins.builtins	This package contains built-in plugins for ObjectGrid core.
com.ibm.websphere.objectgrid.plugins.index	
com.ibm.websphere.objectgrid.plugins.io.annotations	
com.ibm.websphere.objectgrid.	This package has the class MapPermission and class AdminPermission which represents the permissions for to access the ObjectGrid maps and

security	ObjectGrid administration respectively.
com.ibm.websphere.objectgrid.security.config	This package contains the ObjectGrid client security configurations.
com.ibm.websphere.objectgrid.security.plugins	This package contains the interfaces for adding plug-ins to the ObjectGrid security framework and assoicated Exception classes.
com.ibm.websphere.objectgrid.security.plugins.builtins	This package contains the built-in implementation for the security plugins.
com.ibm.websphere.objectgrid.spring	This package holds the Spring specific APIs for ObjectGrid.
com.ibm.websphere.xs.ra	These are the eXtreme Scale Resource Adapter APIs that allows integration with a Java EE application.
com.ibm.websphere.xsa	

Package com.ibm.websphere.objectgrid

These are the main application APIs for the ObjectGrid.

See:

Description

Interface Summary

BackingMap	This is the public interface to the BackingMap.
CatalogDomainInfo	Identifies the configuration attributes of a catalog service domain.
CatalogDomainManager	Provides access to catalog domain configuration information for the current environment.
ClientClusterContext	This interface is a context to represent which cluster/domain the client connected to using one of the `ObjectGridManager.connect` methods.
ClientReplicableMap	**Deprecated.** *The client replicated map function is deprecated in version 8.6.*
IObjectGridException	This interface is used to ensure JDK 1.4 Throwable chaining behavior for all exceptions thrown by ObjectGrid even when an earlier JDK is used (e.g.
JavaMap	This interface is a handle to a named Map.
ObjectGrid	This object is used for creating sessions to the ObjectGrid.
ObjectGridManager	ObjectGridManager is responsible for creating or retrieving local ObjectGrid instances and connecting to distributed ObjectGrid servers.
ObjectMap	This is a handle to a named Map.
PartitionManager	This interface will be used for calculating the proper partition for a given input key.
Session	This interface represents a session container for `ObjectMaps`.
StateManager	The StateManager can be used to retrieve the availability state of an ObjectGrid.
TxID	This interface is an opaque identifier for a transaction.

Class Summary

AvailabilityState	Each shard in a distributed ObjectGrid has an availability state associated with it.
ClientReplicableMap.Mode	Client Replication mode
CopyMode	This class is used to define the "copy" mode when the `setCopyMode` method of the `BackingMap` interface is used.
LockStrate	LockStrategy provides an enumerated type idiom for use on the

gy	`BackingMap.setLockStrategy(LockStrategy)` method.
ObjectGridManagerFactory	This factory class is a high level helper class to get `ObjectGridManager` instances.
TTLType	Every BackingMap in ObjectGrid has a built in timed based evictor that is referred to as "time to live" evictor or TTL evictor.

Enum Summary

ObjectMap.PutMode	Identifies the operation mode of the `ObjectMap.put(Object, Object)`, `ObjectMap.putAll(Map)` , `JavaMap.put(Object, Object)` and `JavaMap.putAll(Map)` methods.
Session.TxCommitProtocol	The commit protocols that can be used to commit the Session's transaction

Exception Summary

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException	This exception is thrown when a method call to the Client/Server TransactionCallback detects the user is attempting to perform a write against multiple maps in different Map Sets, Partition Sets or Replication groups.
ConnectException	This exception is used to indicate that the client was unable to connect to the server
DuplicateKeyException	A DuplicateKeyException exception is thrown if a key cannot be inserted into a BackingMap because an object with the same key already exists.
KeyNotFoundException	Normally, record not found means a null is returned.
LockDeadlockException	This exception is used by the lock manager to indicate that it detected a deadlock.
LockException	A general locking exception indicating something went wrong with locking operations.
LockInternalFailureException	This exception is used by the lock manager to indicate it detected some internal programming error while processing a lock or unlock request.
LockTimeoutException	This exception is used by the lock manager to indicate that the maximum wait time for a lock has been exceeded.
NoActiveTransactionException	An exception indicating there is no active transaction.
ObjectGridException	Base exception class for all checked exceptions thrown by the ObjectGrid product.
ObjectGrid	

RuntimeException	This exception is the base class for all runtime exceptions thrown by the cache.
ReadOnlyException	This exception is thrown when an attempt is made to modifying operations on a read only maps.
ReplicationVotedToRollbackTransactionException	This exception is thrown when a transaction was rolled back because some/all of the replicas failed to apply the transaction when in synchronous replication mode.
SessionNotReentrantException	A Session object can only be used by a single thread concurrently to perform map operations.
TargetNotAvailableException	A `TargetNotAvailableException` indicates the ObjectGrid target is not available.
TransactionAffinityException	This exception is thrown for inflight transaction when server fails over.
TransactionAlreadyActiveException	An exception indicating a transaction is already active for the current session.
TransactionException	A general transaction exception indicating something went wrong with a transaction.
TransactionQuiesceException	This exception is thrown when partition/shard/mapset/replication group/ replication group member/server/cluster/objectgrid is entered quiesce process for various reasons such as shard movement, partition relocation, system update, server shutdown, and others.
TransactionTimeoutException	This exception is thrown when a transaction exceeds the transaction timeout that was specified on the `ObjectGrid` or `Session`.
UnavailableServiceException	This exception is thrown when all servers are dead or when all services are unavailable even though servers are running.
UndefinedMapException	This exception indicates that the map which an application tries to access is not defined in the ObjectGrid.

Package com.ibm.websphere.objectgrid Description

These are the main application APIs for the ObjectGrid. The main interface here is the ObjectGrid interface. A JVM needs to create at least one instance.

Introduction

The WebSphere ObjectGrid is designed as a data caching tier that can be used to hold data from multiple sources and then make it available to the clients of the ObjectGrid. The clients access the data through the ObjectGrid APIs. The ObjectGrid is designed to be able to store large quantities of data.

Programming Tutorial

The following sections show snippets on the usage of the ObjectGrid APIs.

Obtaining a ObjectGrid instance.

The application needs to construct an ObjectGrid reference first. An application can choose to make several ObjectGrid instances. Each instance is independent, however, and has it's own configuration file. For now, use the following code and programmatically initialize it using the setter methods on the ObjectGrid.

```
ObjectGrid objectGrid = new ObjectGridImpl();
```

The instance can then have a Map defined on it using the following snippet:

```
BackingMap bm = objectGrid.defineMap("TABLE1");
```

Again, setter methods on BackingMap allow it to be configured once it's defined.

Working with an ObjectGrid, Sessions

Each thread that wants to access the ObjectGrid must have its own Session instance. The ObjectGrid class has a getSession method that returns one. Once the thread has a Session then it can obtain ObjectMap instances for manipulating data in the ObjectGrid as well as use the begin/commit/rollback methods on the Session to handle transactions.

```
Session session = objectGrid.getSession();
ObjectMap table1 = session.getMap("TABLE1");
session.begin();
MyData d = (MyData)table1.get("key1");
session.commit();
```

Package com.ibm.websphere.objectgrid.client

This package contains the primary interfaces and classes for setting behaviors for ObjectGrid clients for this process.

See:

Description

Interface Summary	
ClientProperties	The set of properties used to define various preference for ObjectGrid clients.

Package com.ibm.websphere.objectgrid.client Description

This package contains the primary interfaces and classes for setting behaviors for ObjectGrid clients for this process.

Overview

The interfaces in this package should not be implemented directly but are used by the `ClientClusterContext` to set default behaviors for application clients for an ObjectGrid instance.

The properties available for use are defined in the `ClientProperties` interface.

There are two ways to configure client properties:

1. Create a properties file named `objectGridClient.properties` and store it in the root of your classpath.
2. Create a properties file on your file system in the directory where the client is started from named objectGridClient.properties.
3. Create a properties file with any path and name and use the following system property to detect it: `-Dcom.ibm.websphere.objectgrid.ClientProperties=<fileName>`
4. Create a properties file with any path and name and set load it programmatically and pass url to ClientClusterContext.getClientProperties(ogname, url).
5. Programmatically define the properties the `ClientProperties` set methods.

In the following example we set the proximity routing defaults for all clients that use this ClientClusterContext:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ClientClusterContext ccc = ogMgr.connect(...);
ClientProperties props = ccc.getClientProperties("myOGName");
props.setPreferLocalHost(true);
props.setPreferLocalProcess(true);
props.setPreferZones(new String[]{"New York", "Texas"});

// The ClientProperites are now applied to the ObjectGrid client connection:
ObjectGrid og=ogMgr.get(ccc, "myOGName");
```

The following example uses a custom client properties file:

```
ClientClusterContext ccc = ogMgr.connect(...);
```

```
    URL clientPropsURL = Thread.currentThread().getContextClassLoader().getResource("etc/myObjectG
ridClient.properties");
    ClientProperties props = ccc.setClientProperties("myOGName", clientPropsURL);

    // The ClientProperites are now applied to the ObjectGrid client connection:
    ObjectGrid og=ogMgr.get(ccc, "myOGName");
```

The following file is an example of a properties file that matches the proceeding API:

```
    preferLocalProcess=true
    preferLocalhost=true
    preferZones=New York,Texas
```

---

com.ibm.websphere.objectgrid.client

Interface ClientProperties

public interface **ClientProperties**

The set of properties used to define various preference for ObjectGrid clients.

See the package summary for details on how to use the ClientProperties class and properties file.

Since:
WAS XD 6.1.0.3, XC10

Field Summary

static String	**CLIENT_PROPS_FILE_PATH_KEY** The system property key to override the location of the client properties file.
static String	**DEFAULTCLIENTPROPERTYFILE** The default name of client property file
static String	**PROP_LISTENER_HOST** Listener host property key for the client properties file.
static St	**PROP_LISTENER_PORT** Listener port property key for the client properties file.

String	
static String	**PROP_MAXIMUM_XIO_NETWORK_THREAD_POOL_SIZE** Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.
static String	**PROP_MAXIMUM_XIO_WORKER_THREAD_POOL_SIZE** Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.
static String	**PROP_MINIMUM_XIO_NETWORK_THREAD_POOL_SIZE** Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.
static String	**PROP_MINIMUM_XIO_WORKER_THREAD_POOL_SIZE** Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.
static String	**PROP_PREFER_LOCAL_HOST** Currently, this property is not used.
static String	**PROP_PREFER_LOCAL_PROCESS** Currently, this property is not used.

[ng](#)	
static [String](#)	**PROP_PREFER_ZONES** Prefer zones property key for the client properties file.
static [String](#)	**PROP_REQUEST_RETRY_TIMEOUT** The requestRetryTimeout which indicates how long to retry a request (in milliseconds).
static [String](#)	**PROP_SHUFFLE_BOOTSTRAP_ADDRESSES** The shuffleBoostrapAddresses property is used to determine if the catalog service grid addresses should be randomized when used by a client when bootstrapping to the grid.
static [String](#)	**PROP_XIO_REQUEST_TIMEOUT** The xioRequestTimeout indicates how long the eXtreme IO transport will wait for cross-process call to complete.
static [String](#)	**PROP_XIO_TIMEOUT** Sets the timeout for server requests using the eXtremeIO transport.

Method Summary

[String](#)	**getListenerHost**() Retrieves the host to be used by the ORB.

int	**getListenerPort**() Retrieves the port to be used by the ORB.
int	**getMaximumXIONetworkThreads**() Gets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.
int	**getMaximumXIOWorkerThreads**() Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.
int	**getMinimumXIONetworkThreads**() Gets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.
int	**getMinimumXIOWorkerThreads**() Retrieves the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.
String[]	**getPreferZones**() Retrieve the preferred zones.
long	**getRequestRetryTimeout**() Retrieves the current request retry timeout.
int	**getXIORequestTimeout**() Gets the current timeout that is allowed for requests to servers over the XIO transport to complete.
int	**getXIOTimeout**() Returns the current timeout for server requests using the XIO transport.
boolean	**isBootStrapListShuffled**() Retrieves the value of the BootStrapListShuffled property.
boolean	**isPreferLocalHost**() This method is reserved for future use.
boolean	**isPreferLocalProcess**() This method is reserved for future use.
void	**setBootStrapListShuffled**(boolean shuffle) Sets the value of the BootStrapListShuffled property.

void	**setMaximumXIONetworkThreads**(int maxThreads) Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.
void	**setMaximumXIOWorkerThreads**(int maxThreads) Retrieves the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.
void	**setMinimumXIONetworkThreads**(int minThreads) Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.
void	**setMinimumXIOWorkerThreads**(int minThreads) Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.
void	**setPreferLocalHost**(boolean localHost) This method is reserved for future use.
void	**setPreferLocalProcess**(boolean localProcess) This method is reserved for future use.
void	**setPreferZones**(String[] zones) Prefer routing to specific zones.
void	**setRequestRetryTimeout**(long requestRetryTimeout) Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions.
void	**setXIORequestTimeout**(int timeout) Sets the current timeout that is allowed for requests to servers over the XIO transport to complete.
void	**setXIOTimeout**(int timeout) Sets the current timeout for server requests using the XIO transport.

Field Detail

DEFAULTCLIENTPROPERTYFILE

static final String **DEFAULTCLIENTPROPERTYFILE**

The default name of client property file

See Also:
Constant Field Values

CLIENT_PROPS_FILE_PATH_KEY

static final String **CLIENT_PROPS_FILE_PATH_KEY**

The system property key to override the location of the client properties file.

Since:
7.0
See Also:
Constant Field Values

PROP_PREFER_LOCAL_PROCESS

`static final String` **`PROP_PREFER_LOCAL_PROCESS`**

Currently, this property is not used. It is reserved for future use.

See Also:
`setPreferLocalProcess(boolean)`, Constant Field Values

PROP_PREFER_LOCAL_HOST

`static final String` **`PROP_PREFER_LOCAL_HOST`**

Currently, this property is not used. It is reserved for future use.

See Also:
`setPreferLocalHost(boolean)`, Constant Field Values

PROP_PREFER_ZONES

`static final String` **`PROP_PREFER_ZONES`**

Prefer zones property key for the client properties file. Each specified zone is separated by a comma in the form: preferZones=ZoneA,ZoneB,ZoneC

See Also:
`setPreferZones(String[])`, Constant Field Values

PROP_REQUEST_RETRY_TIMEOUT

`static final String` **`PROP_REQUEST_RETRY_TIMEOUT`**

The requestRetryTimeout which indicates how long to retry a request (in milliseconds). A 0 indicates that the request should fail fast and skip over in internal retry logic. Exceptions that cannot succeed even if tried again such as DuplicateException will be returned immediately.

Since:
7.0
See Also:
`setRequestRetryTimeout(long)`, Constant Field Values

PROP_XIO_REQUEST_TIMEOUT

`static final String` **`PROP_XIO_REQUEST_TIMEOUT`**

The xioRequestTimeout indicates how long the eXtreme IO transport will wait for cross-process call to complete. The value is expressed in milliseconds. The default value is 30,000 or 30 seconds. When custom tuning client side retry of eXtreme Scale operations, this value will determine how long a single network operation is given, and then the

PROP_REQUEST_RETRY_TIMEOUT / requestRetryTimeout will control how long those individual network operations are retried.

Since:
8.6.0.2, XC10 2.5
See Also:
Constant Field Values

PROP_LISTENER_HOST

`static final String` **`PROP_LISTENER_HOST`**

Listener host property key for the client properties file.

Since:
XS 7.1
See Also:
`getListenerHost()`, Constant Field Values

PROP_LISTENER_PORT

`static final String` **`PROP_LISTENER_PORT`**

Listener port property key for the client properties file.

Since:
XS 7.1
See Also:
`getListenerPort()`, Constant Field Values

PROP_SHUFFLE_BOOTSTRAP_ADDRESSES

`static final String` **`PROP_SHUFFLE_BOOTSTRAP_ADDRESSES`**

The shuffleBoostrapAddresses property is used to determine if the catalog service grid addresses should be randomized when used by a client when bootstrapping to the grid. The default value of the property is true.

Since:
7.1.0.3
See Also:
Constant Field Values

PROP_XIO_TIMEOUT

`static final String` **`PROP_XIO_TIMEOUT`**

Sets the timeout for server requests using the eXtremeIO transport. The timeout is set in seconds. The default xioTimeout for server requests is set to 30 seconds. The valid range is timeout >= 1.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MAXIMUM_XIO_NETWORK_THREAD_POOL_SIZE

static final String **PROP_MAXIMUM_XIO_NETWORK_THREAD_POOL_SIZE**

Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MAXIMUM_XIO_WORKER_THREAD_POOL_SIZE

static final String **PROP_MAXIMUM_XIO_WORKER_THREAD_POOL_SIZE**

Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MINIMUM_XIO_NETWORK_THREAD_POOL_SIZE

static final String **PROP_MINIMUM_XIO_NETWORK_THREAD_POOL_SIZE**

Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

PROP_MINIMUM_XIO_WORKER_THREAD_POOL_SIZE

static final String **PROP_MINIMUM_XIO_WORKER_THREAD_POOL_SIZE**

Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

Method Detail

setPreferZones

void **setPreferZones**(String[] zones)

Prefer routing to specific zones.

When zones are enabled on an ObjectGrid, requests will be routed to the specified zones.

Parameters:
`zones` - array of zone names. If null or an empty array, then requests are routed to all zones.

setPreferLocalProcess

```
void setPreferLocalProcess(boolean localProcess)
```

This method is reserved for future use. Calls to the method will not result in any performed operation.

Parameters:

`localProcess` -

---

setPreferLocalHost

```
void setPreferLocalHost(boolean localHost)
```

This method is reserved for future use. Calls to the method will not result in any performed operation.

Parameters:

`localHost` -

---

getPreferZones

```
String[] getPreferZones()
```

Retrieve the preferred zones.

Returns:

the preferred zones.

---

isPreferLocalProcess

```
boolean isPreferLocalProcess()
```

This method is reserved for future use. The returned value should be ignored by the user.

Returns:

false

---

isPreferLocalHost

```
boolean isPreferLocalHost()
```

This method is reserved for future use. The returned value should be ignored by the user.

Returns:

false

---

setRequestRetryTimeout

```
void setRequestRetryTimeout(long requestRetryTimeout)
```

Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions. A request will timeout when either the request timeout expires or the transaction timeout expires, whichever expires first.

A value of 0 indicates that all requests should fail immediately and avoid any retry logic.

Exceptions that cannot succeed even if tried again such as DuplicateKeyException exceptions will be thrown immediately.

A value of -1 indicates that the request retry timeout is not set, meaning that the request duration is governed by the transaction timeout.

The request retry timeout can be overridden using the `Session.setRequestRetryTimeout(long)` method.

Parameters:

`requestRetryTimeout` - the duration in milliseconds retry a client request, 0 if the request should fail immediately or -1 if the request timeout is not set.

Since:

7.0

See Also:

`Session.setRequestRetryTimeout(long)`, `ObjectGrid.setTxTimeout(int)`

---

getRequestRetryTimeout

`long` **`getRequestRetryTimeout`**`()`

Retrieves the current request retry timeout. Returns -1 if it was not set.

Returns:

requestRetryTimeout in milliseconds, 0 to fail immediately or -1 if not set.

Since:

7.0

---

getListenerHost

`String` **`getListenerHost`**`()`

Retrieves the host to be used by the ORB. The listener host property defaults to 'localhost'. This property can only be set in the client.properties file.

Returns:

The host that the ORB will bind to.

Since:

7.1

---

getListenerPort

`int` **`getListenerPort`**`()`

Retrieves the port to be used by the ORB. The listener port property defaults to the corbaloc port, 2809. This property can only be set in the client.properties file.

Returns:

The port that the ORB will bind to.

Since:

7.1

---

isBootStrapListShuffled

`boolean` **`isBootStrapListShuffled`**`()`

Retrieves the value of the BootStrapListShuffled property.

Returns:
true if the value of BootStrapListeShuffled was set to true. false if the value of BootStrapListeShuffled was set to false.
Since:
7.1.0.3

---

setBootStrapListShuffled

`void` **`setBootStrapListShuffled`**`(boolean shuffle)`

Sets the value of the BootStrapListShuffled property.

Parameters:
`shuffle` - true the bootstrap list will be shuffled providing each client a random distribution of catalog servers to select from. false the first viable address in the list of catalog servers will be used.
Since:
7.1.0.3

---

getMinimumXIOWorkerThreads

`int` **`getMinimumXIOWorkerThreads`**`()`

Retrieves the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Returns:
the mininum number of threads.
Since:
8.6, XC10 2.5

---

setMinimumXIOWorkerThreads

`void` **`setMinimumXIOWorkerThreads`**`(int minThreads)`

Sets the minimum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Parameters:
`minThreads` - the minimum number of threads.
Since:
8.6, XC10 2.5

---

getMaximumXIOWorkerThreads

`int` **`getMaximumXIOWorkerThreads`**`()`

Sets the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Returns:
the maximum number of threads.
Since:
8.6, XC10 2.5

---

setMaximumXIOWorkerThreads

`void` **`setMaximumXIOWorkerThreads`**`(int maxThreads)`

Retrieves the maximum number of threads to allocate in the eXtremeIO transport request processing thread pool.

Parameters:
`maxThreads` - the maximum number of threads.
Since:
8.6, XC10 2.5

---

getMinimumXIONetworkThreads

`int` **`getMinimumXIONetworkThreads`**`()`

Gets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Returns:
the minimum number of threads
Since:
8.6, XC10 2.5

---

setMinimumXIONetworkThreads

`void` **`setMinimumXIONetworkThreads`**`(int minThreads)`

Sets the minimum number of threads to allocate in the eXtremeIO transport network thread pool.

Parameters:
`minThreads` - the minimum number of threads
Since:
8.6, XC10 2.5

---

getMaximumXIONetworkThreads

`int` **`getMaximumXIONetworkThreads`**`()`

Gets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Returns:
the maximum number of threads.
Since:
8.6, XC10 2.5

---

setMaximumXIONetworkThreads

`void` **`setMaximumXIONetworkThreads`**`(int maxThreads)`

Sets the maximum number of threads to allocate in the eXtremeIO transport network thread pool.

Parameters:
`maxThreads` - the maximum number of threads.
Since:
8.6, XC10 2.5

---

getXIOTimeout

`int` **`getXIOTimeout`**`()`

Returns the current timeout for server requests using the XIO transport.

Returns:
the current timeout in seconds
Since:
8.6, XC10 2.5

---

setXIOTimeout

`void` **`setXIOTimeout`**`(int timeout)`

Sets the current timeout for server requests using the XIO transport. The timeout is set in seconds. The valid range is timeout >= 1.

Parameters:
`timeout` - the timeout in seconds
Since:
8.6, XC10 2.5

---

getXIORequestTimeout

`int` **`getXIORequestTimeout`**`()`

Gets the current timeout that is allowed for requests to servers over the XIO transport to complete. The timeout is set in milliseconds.

Since:
8.6.0.2, XC10 2.5

---

setXIORequestTimeout

`void` **`setXIORequestTimeout`**`(int timeout)`

Sets the current timeout that is allowed for requests to servers over the XIO transport to complete. The timeout is set in milliseconds. The valid range is greater than 0.

Parameters:
`timeout` - the timeout in milliseconds
Since:
8.6.0.2, XC10 2.5

Package com.ibm.websphere.objectgrid.config

This package contains the interfaces and a factory class for creating ObjectGrid configuration objects programatically.

See:

Description

Interface Summary

BackingMapConfiguration	A BackingMapConfiguration object can be used to override BackingMap settings on the client side.
ConfigProperty	ConfigProperty can be used to attach properties to a `Plugin`.
ObjectGridConfiguration	An ObjectGridConfiguration object can be used to override ObjectGrid plugins on the client side.
Plugin	This interface represents an ObjectGrid or BackingMap plugin.
PluginType	Every `Plugin` has a PluginType.

Class Summary

ConfigPropertyType	ConfigPropertyType is used to set the type of an attribute on a Plugin.
ObjectGridConfigFactory	This is the configuration factory for ObjectGrid configuration entities.
QueryConfig	This QueryConfig represents a schema configuration for a QueryManager.
QueryMapping	A QueryMapping maps a Java class to a BackingMap and allows a map to be included in a query.
QueryRelationship	A QueryRelationship represents a relationship between two BackingMap value classes.

Exception Summary

ObjectGridConfigurationException	Thrown when a problem with the current configuration is found.

Package com.ibm.websphere.objectgrid.config Description

This package contains the interfaces and a factory class for creating ObjectGrid configuration objects programatically. The main use of this is by the objectgrid client to override serverside configuration.

Overview

ObjectGridManagerFactory has static methods to create the configuration objects. Using these configuration objects in conjuction with ObjectGridManager methods

- setOverrideObjectGridConfigurations(Map)
- putOverrideObjectGridConfigurations(String, List)

to override client configuration, before connecting to the objectgrid server.

com.ibm.websphere.objectgrid.config

Class QueryRelationship

java.lang.Object
  com.ibm.websphere.objectgrid.config.QueryRelationship

All Implemented Interfaces:

Serializable

```
public class QueryRelationship
extends Object
implements Serializable
```

A QueryRelationship represents a relationship between two BackingMap value classes. A BackingMap must have one class type defined in the value part of the map. A relationship can be established between two maps by mapping the source and target map's value classes.

The cardinality of the relationship is automatically determined by the type of the attribute field.

A relationship requires two classes, a relationship field, and optionally an inverse realtionship field.

For example: Two entities; Department and Employee, have the following bi-directional relationship:

1. One department has many employees. the collection field in the Department class is "emps"
2. An employee belongs to one department

```
public class Department {
    private int id;
    private Collection emps;

    public void setEmps(Collection emps) {
        this.emps = emps;
    }

    public Collection getEmps() {
        return emps;
    }
    ...
}

public class Employee {
    private int id;
    private Department dept;

    public void setDept(Department dept) {
        this.dept = dept;
    }

    public Department getDept() {
        return dept;
    }
```

```
}
```

Use the following method call to establish this bi-directional relationship.

```
queryConfig.addRelationship(new QueryRelationship(
    Department.class.getName(), Employee.class.getName(), "emps", "dept"));
```

Since:

WAS XD 6.1, XC10

See Also:

Serialized Form

Constructor Summary

QueryRelationship(String sourceClass, String targetClass, String relationshipField, String invRelationshipField) Constructor for creating a QueryRelationship instance.

Method Summary

boolean	**equals**(Object o)
String	**getInvRelationshipField**() Retrieve the inverse relationship attribute name of a bi-directional relationship.
String	**getRelationshipField**() Retrieve the name of the attribute in the source class that references the key of the target class.
String	**getSourceClass**() Retrieve the name of the class representing the source of a relationship.
String	**getTargetClass**() Retrieve the name of the class representing the target of a relationship.
int	**hashCode**()
void	**setInvRelationshipField**(String invRelationshipField) Set the inverse relationship attribute name of a bi-directional relationship.

void	**setRelationshipField**(String relationshipField) Set the name of the attribute in the source class that references the key of the target class.
void	**setSourceClass**(String sourceClass) Set the name of the class representing the source of a relationship.
void	**setTargetClass**(String targetClass) Set the name of the class representing the target of a relationship.
String	**toString**()

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

QueryRelationship

```
public QueryRelationship(String sourceClass,
                         String targetClass,
                         String relationshipField,
                         String invRelationshipField)
```

Constructor for creating a QueryRelationship instance.

The sourceClass, targetClass, and relationshipField must not be null.

Parameters:

`sourceClass` - the source class of the relationship
`targetClass` - the target class of the relationship
`relationshipField` - the attribute in the source class that references the key of the target class.
`invRelationshipField` - the attribute in the target class that references the key of the source class. This value is null if a bi-directional relationship does not exist.

Method Detail

getInvRelationshipField

```
public String getInvRelationshipField()
```

Retrieve the inverse relationship attribute name of a bi-directional relationship.

Returns:

the attribute name of the inverse side of a bi-directional relationship or null if the relationship is uni-directional.

---

setInvRelationshipField

public void **setInvRelationshipField**(String invRelationshipField)

Set the inverse relationship attribute name of a bi-directional relationship.

Parameters:

`invRelationshipField` - the attribute name of the inverse side of a bi-directional relationship or null if the relationship is uni-directional.

---

getRelationshipField

public String **getRelationshipField**()

Retrieve the name of the attribute in the source class that references the key of the target class.

Returns:

the name of the relationship attribute.

---

setRelationshipField

public void **setRelationshipField**(String relationshipField)

Set the name of the attribute in the source class that references the key of the target class.

Parameters:

`relationshipField` - the name of the relationship attribute.

---

getSourceClass

public String **getSourceClass**()

Retrieve the name of the class representing the source of a relationship.

Returns:

the source class

---

setSourceClass

public void **setSourceClass**(String sourceClass)

Set the name of the class representing the source of a relationship.

Parameters:

`sourceClass` - the source class

---

getTargetClass

public String **getTargetClass**()

Retrieve the name of the class representing the target of a relationship.

Returns:

the target class

---

setTargetClass

public void **setTargetClass**(String targetClass)

Set the name of the class representing the target of a relationship.

Parameters:

`targetClass` - the target class

---

equals

public boolean **equals**(Object o)

Overrides:

equals in class Object

See Also:

Object.equals(java.lang.Object)

---

hashCode

public int **hashCode**()

Overrides:

hashCode in class Object

See Also:

Object.hashCode()

---

toString

public String **toString**()

Overrides:

toString in class Object

---

com.ibm.websphere.objectgrid.config

Class QueryMapping

java.lang.Object
  com.ibm.websphere.objectgrid.config.QueryMapping

All Implemented Interfaces:
Serializable

```
public class QueryMapping
extends Object
implements Serializable
```

A QueryMapping maps a Java class to a BackingMap and allows a map to be included in a query. It also indicates whether the query engine should use a getter method or direct field access to access fields in the value class.

For example, class Department is the value class that is stored in the "DepartmentMap" BackingMap and the key is an Integer.

```
public class Department {
    private int id;
    private Collection emps;

    public void setEmps(Collection emps) {
        this.emps = emps;
    }

    public Collection getEmps() {
        return emps;
    }
    ...
}
```

The QueryMapping would be created as follows:

```
...
QueryConfig queryConfig = new QueryConfig();
queryConfig.addMapping(new QueryMapping(
    "DepartmentMap", Department.class.getName(), "id", QueryMapping.PROPERTY_ACCESS)
objectGrid.setQueryConfig(queryConfig);
...
```

Since:
WAS XD 6.1, XC10
See Also:
Serialized Form

Field Summary

s t a

static int	**FIELD_ACCESS** This constant indicates to use direct field access to read the field values
static int	**PROPERTY_ACCESS** This constant indicates to use JavaBean property-style get methods to read the field values from the Java object stored in the BackingMap.

Constructor Summary

QueryMapping() Default constructor.
QueryMapping(String mapName, String valueClass, String primaryKeyField) Constructor for creating a basic QueryMapping instance with a default access type of PROPERTY_ACCESS.
QueryMapping(String mapName, String valueClass, String primaryKeyField, int accessType) Constructor for creating a QueryMapping instance.

Method Summary

boolean	**equals**(Object o)
int	**getAccessType**() Retrieve the method in which the query engine will access the value class object stored in the BackingMap.
String	**getMapName**() Retrieve the BackingMap name associated with this mapping.
String	**getPrimaryKeyField**() Retrieve the name of the attribute in the value class object that is also the primary key of the BackingMap.
String	**getValueClass**() Retrieve the type of object that is stored in the BackingMap.
int	**hashCode**()

void	**setAccessType**(int accessType) Set the method in which the query engine will access the value class object stored in the BackingMap.
void	**setMapName**(String mapName) Set the BackingMap name associated with this mapping.
void	**setPrimaryKeyField**(String primaryKeyField) Set the name of the attribute in the value class object that is also the primary key of the BackingMap.
void	**setValueClass**(String valueClass) Set the type of object that is stored in the BackingMap.
String	**toString**()

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

FIELD_ACCESS

public static final int **FIELD_ACCESS**

This constant indicates to use direct field access to read the field values

See Also:
setAccessType(int), getAccessType(), Constant Field Values

PROPERTY_ACCESS

public static final int **PROPERTY_ACCESS**

This constant indicates to use JavaBean property-style get methods to read the field values from the Java object stored in the BackingMap. PROPERTY_ACCESS is the default.

See Also:
setAccessType(int), getAccessType(), Constant Field Values

Constructor Detail

QueryMapping

public **QueryMapping**()

Default constructor.

QueryMapping

```
public QueryMapping(String mapName,
                    String valueClass,
                    String primaryKeyField)
```

Constructor for creating a basic QueryMapping instance with a default access type of PROPERTY_ACCESS.

The mapName and valueClass must not be null.

Parameters:

`mapName` - the name of the BackingMap to map
`valueClass` - the class of object stored in the BackingMap's value.
`primaryKeyField` - the optional name of the primary key field of the class.

QueryMapping

```
public QueryMapping(String mapName,
                    String valueClass,
                    String primaryKeyField,
                    int accessType)
```

Constructor for creating a QueryMapping instance. The mapName and valueClass must not be null.

Parameters:

`mapName` - the name of the BackingMap to map
`valueClass` - the class of object stored in the BackingMap's value.
`primaryKeyField` - the optional name of the primary key field of the class.
`accessType` - the method (PROPERTY_ACCESS or FIELD_ACCESS) in which the query engine will access the persistent data in the value object.

Method Detail

getMapName

```
public String getMapName()
```

Retrieve the BackingMap name associated with this mapping.

Returns:

the BackingMap name.

---

setMapName

```
public void setMapName(String mapName)
```

Set the BackingMap name associated with this mapping.

---

getValueClass

```
public String getValueClass()
```

Retrieve the type of object that is stored in the BackingMap.

Returns:

the object type that is stored in the BackingMap's value.

---

setValueClass

`public void **setValueClass**(String valueClass)`

Set the type of object that is stored in the BackingMap.

---

getAccessType

`public int **getAccessType**()`

Retrieve the method in which the query engine will access the value class object stored in the BackingMap.

Returns:
Returns the accessType.
See Also:
PROPERTY_ACCESS, FIELD_ACCESS

---

setAccessType

`public void **setAccessType**(int accessType)`

Set the method in which the query engine will access the value class object stored in the BackingMap.

Parameters:
`accessType` - the accessType.
See Also:
PROPERTY_ACCESS, FIELD_ACCESS

---

getPrimaryKeyField

`public String **getPrimaryKeyField**()`

Retrieve the name of the attribute in the value class object that is also the primary key of the BackingMap.

This value is optional.

Returns:
the primaryKeyField.

---

setPrimaryKeyField

`public void **setPrimaryKeyField**(String primaryKeyField)`

Set the name of the attribute in the value class object that is also the primary key of the BackingMap.

Parameters:
`primaryKeyField` - the name of the primary key attribute or null if not set.

---

equals

`public boolean **equals**(Object o)`

Overrides:

equals in class Object

See Also:

Object.equals(java.lang.Object)

---

hashCode

public int **hashCode**()

Overrides:

hashCode in class Object

See Also:

Object.hashCode()

---

toString

public String **toString**()

Overrides:

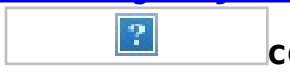
toString in class Object

---

com.ibm.websphere.objectgrid.config

Class QueryConfig

java.lang.Object



com.ibm.websphere.objectgrid.config.QueryConfig

All Implemented Interfaces:

Serializable

```
public class QueryConfig
extends Object
implements Serializable
```

This QueryConfig represents a schema configuration for a QueryManager. A QueryConfig object contains a set of QueryMapping objects and a set of QueryRelationship objects.

Users use addQuerySchema method to add a QueryMapping object and use addRelationship method to add a QueryRelationship object into this QueryConfig object.

Since:

WAS XD 6.1, XC10

See Also:

ObjectGrid.setQueryConfig(QueryConfig), QueryMapping, QueryRelationship, Serialized Form

---

Constructor Summary

QueryConfig()	Default constructor.

Method Summary

void	**addQueryMapping**(QueryMapping mapping) Add a QueryMapping to this QueryConfig
void	**addQueryRelationship**(QueryRelationship relation) Add a QueryRelationship to this QueryConfig
QueryMapping[]	**getQueryMappings**() Retrieve all added QueryMappings

QueryRelationship[]	**getQueryRelationships**() Retrieve all added QueryRelationships

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

QueryConfig

```
public QueryConfig()
```

Default constructor.

Method Detail

addQueryMapping

```
public void addQueryMapping(QueryMapping mapping)
```

Add a QueryMapping to this QueryConfig

Parameters:

`mapping` - the QueryMapping to add.

---

addQueryRelationship

```
public void addQueryRelationship(QueryRelationship relation)
```

Add a QueryRelationship to this QueryConfig

Parameters:

`relation` - the QueryRelationship to add.

---

getQueryMappings

```
public QueryMapping[] getQueryMappings()
```

Retrieve all added QueryMappings

Returns:

array of QueryMapping

---

getQueryRelationships

public QueryRelationship[] **getQueryRelationships**()

Retrieve all added QueryRelationships

Returns:

array of QueryRelationship

---

com.ibm.websphere.objectgrid.config

Interface PluginType

All Superinterfaces:

Serializable

```
public interface PluginType
extends Serializable
```

Every `Plugin` has a PluginType. The PluginType is specified during Plugin creation.

`ObjectGridConfiguration` objects support the following PluginTypes for overriding client-side ObjectGrid plugins:

- `PluginType.TRANSACTION_CALLBACK`
- `PluginType.OBJECTGRID_EVENT_LISTENER`
- `PluginType.COLLISION_ARBITER`

`BackingMapConfiguration` objects support the following PluginTypes for overriding client-side BackingMap plugins:

- `PluginType.EVICTOR`
- `PluginType.MAP_EVENT_LISTENER`

Since:

WAS XD 6.0.1.2, XC10

See Also:

`ObjectGridConfigFactory.createPlugin(PluginType, String)`

Field Summary

static PluginType	**COLLISION_ARBITER** `PluginType.COLLISION_ARBITER` can be used to attach a `CollisionArbiter` plugin to an ObjectGrid.
static PluginType	**EVICTOR** `PluginType.EVICTOR` can be used to attach an `Evictor` to a BackingMap.

[gInType](#)	
static [PluginType](#)	**MAP_EVENT_LISTENER** `PluginType.MAP_EVENT_LISTENER` can be used to attach a [MapEventListener](#) to a BackingMap.
static [PluginType](#)	**MAP_LIFECYCLE_LISTENER** `PluginType.MAP_LIFECYCLE_LISTENER` can be used to attach a `BackingMapLifecycleListener` plugin to an BackingMap
static [PluginType](#)	**MAP_SERIALIZER_PLUGIN** `PluginType.MAP_SERIALIZER_PLUGIN` can be used to attach a `MapSerializerPlugin` to a BackingMap to serialize keys.
static [PluginType](#)	**OBJECTGRID_EVENT_LISTENER** `PluginType.OBJECTGRID_EVENT_LISTENER` can be used to attach an [ObjectGridEventListener](#) plugin to an ObjectGrid
stat	

static PluginType	**OBJECTGRID_LIFECYCLE_LISTENER** PluginType.OBJECTGRID_LIFECYCLE_LISTENER can be used to attach an ObjectGridLifecycleListener plugin to an ObjectGrid
static PluginType	**TRANSACTION_CALLBACK** PluginType.TRANSACTION_CALLBACK can be used to attach a TransactionCallback plugin to an ObjectGrid.

Field Detail

OBJECTGRID_EVENT_LISTENER

static final PluginType **OBJECTGRID_EVENT_LISTENER**

PluginType.OBJECTGRID_EVENT_LISTENER can be used to attach an ObjectGridEventListener plugin to an ObjectGrid

---

TRANSACTION_CALLBACK

static final PluginType **TRANSACTION_CALLBACK**

PluginType.TRANSACTION_CALLBACK can be used to attach a TransactionCallback plugin to an ObjectGrid.

COLLISION_ARBITER

static final PluginType **COLLISION_ARBITER**

PluginType.COLLISION_ARBITER can be used to attach a CollisionArbiter plugin to an ObjectGrid.

OBJECTGRID_LIFECYCLE_LISTENER

static final PluginType **OBJECTGRID_LIFECYCLE_LISTENER**

PluginType.OBJECTGRID_LIFECYCLE_LISTENER can be used to attach an ObjectGridLifecycleListener plugin to an ObjectGrid

Since:
7.1.1

EVICTOR

static final PluginType **EVICTOR**

PluginType.EVICTOR can be used to attach an Evictor to a BackingMap.

MAP_EVENT_LISTENER

static final PluginType **MAP_EVENT_LISTENER**

PluginType.MAP_EVENT_LISTENER can be used to attach a MapEventListener to a BackingMap.

MAP_SERIALIZER_PLUGIN

static final PluginType **MAP_SERIALIZER_PLUGIN**

PluginType.MAP_SERIALIZER_PLUGIN can be used to attach a MapSerializerPlugin to a BackingMap to serialize keys.

Since:
7.1.1

MAP_LIFECYCLE_LISTENER

static final PluginType **MAP_LIFECYCLE_LISTENER**

PluginType.MAP_LIFECYCLE_LISTENER can be used to attach a BackingMapLifecycleListener plugin to an BackingMap

Since:
7.1.1

---

com.ibm.websphere.objectgrid.config

Interface Plugin

All Superinterfaces:

Serializable

```
public interface Plugin
extends Serializable
```

This interface represents an ObjectGrid or BackingMap plugin. An `ObjectGridConfiguration` object supports the following Plugins:

- `PluginType#OBJECTGRID_EVENT_LISTENER`
- `PluginType#TRANSACTION_CALLBACK`

A `BackingMapConfiguration` object supports the following Plugins:

- `PluginType#EVICTOR`
- `PluginType#MAP_EVENT_LISTENER`

A Plugin object has following attributes:

- pluginType: the support PluginTypes are listed above
- className: the plugin implementation class name. This class name will be used to construct the class object. A default constructor must be implemented.
- configProperties: the JavaBean properties for this plugin implementation object.

A plugin object can be created by using `ObjectGridConfigFactory.createPlugin(PluginType, String)` `method. Please refer to ObjectGridConfigFactory for detailed example.`

Since:

WAS XD 6.0.1.2, XC10

Method Summary

void	**addConfigProperty**(ConfigProperty configProp) Add a ConfigProperty to this object.
String	**getClassName**() Get the String representation of the class name of this Plugin
List	**getConfigProperties**() Get the ConfigProperty objects that have been set on this object.
Str	**getOSGiService**()

String	Get the OSGi service name configured for this Plugin.
PluginType	**getPluginType**() Get the PluginType for this Plugin.
void	**setClassName**(String className) The class name that is set must be an implementor of the PluginTypeImpl for this Plugin.
void	**setConfigProperties**(List configPropList) Set the ConfigProperties for this object
void	**setOSGiService**(String osgiService) Set the OSGi service name configured for this Plugin.
void	**setPluginType**(PluginType pluginType) Set the PluginType for this Plugin.

Method Detail

addConfigProperty

void **addConfigProperty**(ConfigProperty configProp)

Add a ConfigProperty to this object.

Parameters:
configProp - - ConfigProperty to add to this object

---

setConfigProperties

void **setConfigProperties**(List configPropList)

Set the ConfigProperties for this object

Parameters:
configPropList -

---

getConfigProperties

List **getConfigProperties**()

Get the ConfigProperty objects that have been set on this object.

Returns:
a List of the ConfigProperty objects that have been added to this object.
See Also:
ConfigProperty

---

getPluginType

PluginType **getPluginType**()

Get the PluginType for this Plugin.

Returns:
the PluginType for this Plugin

---

setPluginType

void **setPluginType**(PluginType pluginType)

Set the PluginType for this Plugin.

The ObjectGridConfiguration plugins include

- PluginType.TRANSACTION_CALLBACK
- PluginType.OBJECTGRID_EVENT_LISTENER

The BackingMapConfiguration plugins include

- PluginType.EVICTOR
- PluginType.MAP_EVENT_LISTENER

Parameters:
pluginType -

---

getClassName

String **getClassName**()

Get the String representation of the class name of this Plugin

Returns:
the String representation of the class name

---

setClassName

void **setClassName**(String className)

The class name that is set must be an implementor of the PluginTypeImpl for this Plugin. For example, if the type of this Plugin is PluginType.EVICTOR, then the className must be an implementor of the com.ibm.websphere.objectgrid.plugins.Evictor interface.

Parameters:
className - - the class name of the Class that implements the PluginType

---

getOSGiService

String **getOSGiService**()

Get the OSGi service name configured for this Plugin. If an OSGi service name is configured for this Plugin, the className configured for this Plugin is ignored.

Returns:
the OSGi service name configured for this Plugin.
Since:
7.1.1

---

setOSGiService

void **setOSGiService**(String osgiService)

Set the OSGi service name configured for this Plugin. If an OSGi service name is configured for this Plugin, the className configured for this Plugin is ignored.

Parameters:
osgiService - the OSGi service name configured for this Plugin.
Since:
7.1.1

com.ibm.websphere.objectgrid.config

Class ObjectGridConfigurationException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.config.ObjectGridConfigurationException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **ObjectGridConfigurationException**
extends ObjectGridException

Thrown when a problem with the current configuration is found. This exception may be thrown when the configuration specified in the deployment policy, ObjectGrid descriptor or security descriptor is incorrect.

Since:

7.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridConfigurationException() Constructs a new ObjectGridConfigurationException with null as its detail message.
ObjectGridConfigurationException(String message) Constructs a new ObjectGridConfigurationException with the specified detail message.
ObjectGridConfigurationException(String message, Throwable cause) Constructs a new ObjectGridConfigurationException with the specified detail message and cause.
ObjectGridConfigurationException(Throwable cause) Constructs a new ObjectGridConfigurationException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridConfigurationException

public **ObjectGridConfigurationException**()

Constructs a new ObjectGridConfigurationException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

ObjectGridConfigurationException

public **ObjectGridConfigurationException**(String message)

Constructs a new ObjectGridConfigurationException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

ObjectGridConfigurationException

public **ObjectGridConfigurationException**(Throwable cause)

Constructs a new ObjectGridConfigurationException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ObjectGridConfigurationException that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

ObjectGridConfigurationException

public **ObjectGridConfigurationException**(String message,
Throwable cause)

Constructs a new ObjectGridConfigurationException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in

this ObjectGridConfigurationException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`

---

com.ibm.websphere.objectgrid.config

Interface ObjectGridConfiguration

public interface **ObjectGridConfiguration**

An ObjectGridConfiguration object can be used to override ObjectGrid plugins on the client side. The `com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener` and the `com.ibm.websphere.objectgrid.plugins.TransactionCallback` Plugins can be overridden.

Since:

WAS XD 6.0.1.2, XC10

See Also:

ObjectGridEventListener, TransactionCallback

Method Summary

void	**addBackingMapConfiguration**(BackingMapConfiguration backingMapConfiguration) Add a BackingMapConfiguration to this ObjectGridConfiguration.
void	**addPlugin**(Plugin plugin) Add a `Plugin` to this ObjectGridConfiguration.
List	**getBackingMapConfigurations**() Get the List of BackingMapConfiguration objects that are attached to this ObjectGridConfiguration object
String	**getName**() Get the name of this ObjectGridConfiguration
List	**getPlugins**() Get the Plugins that have been attached to this ObjectGridConfiguration.
int	**getTxIsolation**() Retrieves the default transaction isolation level.
int	**getTxTimeout**() Gets the transaction timeout.
void	**setBackingMapConfigurations**(List backingMapConfigList) Set the BackingMapConfiguration objects for this ObjectGridConfiguration.

void	**setPlugins**(List pluginList) Set the Plugins for this ObjectGridConfiguration.
void	**setTxIsolation**(int level) Sets the default transaction isolation level for all sessions created by the ObjectGrid.
void	**setTxTimeout**(int seconds) Sets the transaction timeout

Method Detail

getName

String **getName**()

Get the name of this ObjectGridConfiguration

Returns:
the name of this ObjectGridConfiguration

---

addBackingMapConfiguration

void **addBackingMapConfiguration**(BackingMapConfiguration backingMapConfiguration)

Add a BackingMapConfiguration to this ObjectGridConfiguration.

Parameters:
`backingMapConfiguration` -

---

setBackingMapConfigurations

void **setBackingMapConfigurations**(List backingMapConfigList)

Set the BackingMapConfiguration objects for this ObjectGridConfiguration. Any BackingMapConfiguration objects that were previously attached to this ObjectGridConfiguration object will be overridden.

Parameters:
`backingMapConfigList` - - A List of `BackingMapConfiguration` objects.
See Also:
BackingMapConfiguration

---

getBackingMapConfigurations

List **getBackingMapConfigurations**()

Get the List of BackingMapConfiguration objects that are attached to this ObjectGridConfiguration object

Returns:
a List of `BackingMapConfiguration` objects
See Also:

BackingMapConfiguration

---

addPlugin

`void` **`addPlugin`**`(`Plugin `plugin)`

Add a `Plugin` to this ObjectGridConfiguration. The Plugins that can be overridden on a client-side ObjectGrid are `com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener` and `com.ibm.websphere.objectgrid.plugins.TransactionCallback`.

Parameters:

`plugin` -

See Also:

setPlugins(List), Plugin

---

setPlugins

`void` **`setPlugins`**`(`List `pluginList)`

Set the Plugins for this ObjectGridConfiguration. Any Plugins that were previously attached to this ObjectGridConfiguration object will be overridden.

Parameters:

`pluginList` - - a List of Plugins

See Also:

addPlugin(Plugin), Plugin

---

getPlugins

List **`getPlugins`**`()`

Get the Plugins that have been attached to this ObjectGridConfiguration.

Returns:

a List of `Plugin` objects

See Also:

Plugin

---

setTxTimeout

`void` **`setTxTimeout`**`(int seconds)`

Sets the transaction timeout

Parameters:

`seconds` -

Since:

7.1.0.3

See Also:

ObjectGrid.setTxTimeout(int)

---

getTxTimeout

`int` **`getTxTimeout`**`()`

Gets the transaction timeout. The value is in seconds.

Returns:
the transaction timeout in seconds
Since:
7.1.0.3
See Also:
ObjectGrid.getTxTimeout()

---

setTxIsolation

`void` **`setTxIsolation`**`(int level)`

Sets the default transaction isolation level for all sessions created by the ObjectGrid. The constants defined in the `Session` interface are the possible transaction isolation levels. The default is Session.TRANSACTION_REPEATABLE_READ.

Parameters:
`level` - one of the following `Session` constants: Session.TRANSACTION_READ_UNCOMMITTED, Session.TRANSACTION_READ_COMMITTED or Session.TRANSACTION_REPEATABLE_READ or 0 if the TransactionIsolation should not be set.
Since:
7.1.1

---

getTxIsolation

`int` **`getTxIsolation`**`()`

Retrieves the default transaction isolation level.

Returns:
the current transaction isolation level.
Since:
7.1.1
See Also:
setTxIsolation(int)

---

com.ibm.websphere.objectgrid.config

Class ObjectGridConfigFactory

java.lang.Object

com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory

---

public final class **ObjectGridConfigFactory**
extends Object

This is the configuration factory for ObjectGrid configuration entities. Users are expected to use static methods of this factory to create ObjectGrid configuration objects.

Here is a list of static methods used to create configuration objects:

- `createObjectGridConfiguration(String)`: create an ObjectGridConfiguration object
- `createBackingMapConfiguration(String)`: create a `BackingMapConfiguration` object by passing a backing map name
- `createConfigProperty(ConfigPropertyType, String, String)`: create a `ConfigProperty` object
- `createPlugin(PluginType, String)`: create a `Plugin` object

Below is an example of creating an ObjectGrid configuration. A Plugin is added to the `ObjectGridConfiguration` object: `com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener` plugin.

A `BackingMapConfiguration` called "myBackingMap" is then created and added to the `ObjectGridConfiguration`. This `BackingMapConfiguration` also has an Evictor Plugin configured.

Once the ObjectGridConfiguration object has been created , it can be used to call either of these methods

- `com.ibm.websphere.objectgrid.ObjectGridManager.putOverrideObjectGridConfigurations(String, List)`
- `com.ibm.websphere.objectgrid.ObjectGridManager.setOverrideObjectGridConfigurations(Map)`

to set configuration objects, prior to connecting.

```
// Create an ObjectGridConfiguration object
ObjectGridConfiguration ogConfig = ObjectGridConfigFactory.createObjectGridConfiguration(ogName);

// create ObjectGridEventListener plugin
 Plugin eventListener = ObjectGridConfigFactory.createPlugin(PluginType.OBJECTGRID_EVENT_LISTENER,
"com.ibm.test.MyOgEventListener");

 // Add plugin to ObjectGridConfiguration object
 ogConfig.addPlugin(eventListener);

 // Create a BackingMapConfiguration object
 BackingMapConfiguration bmConfig = ObjectGridConfigFactory.createBackingMapConfiguration("mybacki
ngMap");

 // Add BackingMapConfiguration object to ObjectGridConfiguration object
 ogConfig.addBackingMapConfiguration(bmConfig);

 // Set the number of buckets to 1000
 bmConfig.setNumberOfBuckets(1000);
```

```
// Create a Evictor plugin for this backing map.
Plugin evictor = ObjectGridConfigFactory.createPlugin(
BackingMapConfiguration.PLUGIN_EVICTOR,
com.acme.myEvictorImpl.class.getName());

// add Evictor Plugin to the BackingMapConfiguration
bmConfig.addPlugin(evictor);

// Create a ConfigProperty for the Evictor plugin
ConfigProperty sizeProperty = ObjectGridConfigFactory.createConfigProperty(
  ConfigPropertyType.INT_PRIM, "size", "153");

// Add the ConfigProperty to the Evictor plugin
evictor.setConfigProperty(sizeProperty);
```

Since:

WAS XD 6.0.1.2, XC10

See Also:

ObjectGridConfiguration, BackingMapConfiguration, Plugin, ConfigProperty

---

Constructor Summary

ObjectGridConfigFactory()

Method Summary

static BackingMapConfiguration	**createBackingMapConfiguration**(String backingMapConfigName) Create a BackingMapConfiguration object
static ConfigProperty	**createConfigProperty**(ConfigPropertyType configPropType, String name, String value) Create a ConfigPropety object for use on a Plugin.

Property	
static ObjectGridConfiguration	**createObjectGridConfiguration**(String objectGridConfigName) Create an ObjectGridConfiguration object.
static Plugin	**createOSGiServicePlugin**(PluginType pluginType, String osgiServiceName) Create an OSGi Service Plugin for a BackingMapConfiguration or an ObjectGridConfiguration
static Plugin	**createPlugin**(PluginType pluginType, String className) Create a Plugin for a BackingMapConfiguration or an ObjectGridConfiguration

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridConfigFactory

public **ObjectGridConfigFactory**()

Method Detail

createPlugin

```
public static Plugin createPlugin(PluginType pluginType,
                                   String className)
```

Create a Plugin for a BackingMapConfiguration or an ObjectGridConfiguration

Parameters:

`pluginType` - the PluginType.
`className` - of the Plugin implementation class to instantiate

Returns:

a Plugin instance

See Also:

PluginType.OBJECTGRID_EVENT_LISTENER, PluginType.TRANSACTION_CALLBACK, PluginType.COLLISION_ARBITER, PluginType.EVICTOR, PluginType.MAP_EVENT_LISTENER, PluginType.OBJECTGRID_LIFECYCLE_LISTENER, PluginType.MAP_LIFECYCLE_LISTENER

---

createObjectGridConfiguration

```
public static ObjectGridConfiguration createObjectGridConfiguration(String objectGridConfigName)
```

Create an ObjectGridConfiguration object.

Parameters:

`objectGridConfigName` - the name that will be assigned to this ObjectGridConfiguration object

Returns:

the ObjectGridConfiguration object

---

createBackingMapConfiguration

```
public static BackingMapConfiguration createBackingMapConfiguration(String backingMapConfigName)
```

Create a BackingMapConfiguration object

Parameters:

`backingMapConfigName` - the name to assign to this BackingMapConfiguration

Returns:

the BackingMapConfiguration object

---

createConfigProperty

```
public static ConfigProperty createConfigProperty(ConfigPropertyType configPropType,
                                                   String name,
                                                   String value)
```

Create a ConfigPropety object for use on a Plugin.

The Plugin should have a set method that corresponds to the name of this ConfigProperty. The method must accept a parameter of the ConfigPropertyType that is specified on this ConfigProperty. For example, if the name of this ConfigProperty is set to "size", and the type is `ConfigPropertyType.INT_PRIM`, then the Plugin must have the method setSize(int). The value of the ConfigProperty will be passed to the setter of the Plugin when an ObjectGrid is created based on this configuration.

Parameters:

`configPropType` - ConfigPropertyType of the ConfigProperty. Part of the set method's signature, the type of parameter the set method requires. Valid
`name` - of the ConfigProperty. It must correspond to the name of a set method on the Plugin.
`value` - of the ConfigProperty. This value will be passed to the set method on the Plugin.

Returns:

the ConfigProperty object

See Also:

ConfigPropertyType.STRING_JAVA_LANG, ConfigPropertyType.BOOLEAN_JAVA_LANG, ConfigPropertyType.BOOLEAN_PRIM, ConfigPropertyType.BYTE_JAVA_LANG, ConfigPropertyType.BYTE_PRIM, ConfigPropertyType.CHARACTER_JAVA_LANG, ConfigPropertyType.CHAR_PRIM, ConfigPropertyType.DOUBLE_JAVA_LANG, ConfigPropertyType.DOUBLE_PRIM, ConfigPropertyType.FLOAT_JAVA_LANG, ConfigPropertyType.FLOAT_PRIM, ConfigPropertyType.INTEGER_JAVA_LANG, ConfigPropertyType.INT_PRIM, ConfigPropertyType.LONG_JAVA_LANG, ConfigPropertyType.LONG_PRIM, ConfigPropertyType.SHORT_JAVA_LANG

---

createOSGiServicePlugin

```
public static Plugin createOSGiServicePlugin(PluginType pluginType,
                                              String osgiServiceName)
```

Create an OSGi Service Plugin for a BackingMapConfiguration or an ObjectGridConfiguration

Parameters:

`pluginType` - the PluginType.
`osgiServiceName` - the OSGi service name

Returns:

a Plugin instance

Since:

7.1.1

See Also:

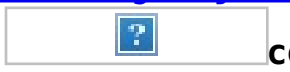
PluginType.OBJECTGRID_EVENT_LISTENER, PluginType.TRANSACTION_CALLBACK, PluginType.COLLISION_ARBITER, PluginType.EVICTOR, PluginType.MAP_EVENT_LISTENER, PluginType.OBJECTGRID_LIFECYCLE_LISTENER, PluginType.MAP_LIFECYCLE_LISTENER

---

com.ibm.websphere.objectgrid.config

Class ConfigPropertyType

java.lang.Object



com.ibm.websphere.objectgrid.config.ConfigPropertyType

All Implemented Interfaces:

Serializable

```
public final class ConfigPropertyType
extends Object
implements Serializable
```

ConfigPropertyType is used to set the type of an attribute on a Plugin. The Java primitives, their java.lang counterparts, and java.lang.String are the supported types.

Since:

WAS XD 6.0.1.2, XC10

See Also:

ObjectGridConfigFactory.createConfigProperty(ConfigPropertyType, String, String), Plugin.setPluginType(PluginType), Serialized Form

---

Field Summary

static ConfigPropertyType	**BOOLEAN_JAVA_LANG** ConfigPropertyType.BOOLEAN_JAVA_LANG can be used to set a property of type java.lang.Boolean on a plugin.
static Con	

static ConfigPropertyType	**BOOLEAN_PRIM** ConfigPropertyType.BOOLEAN_PRIM can be used to set a property of type `boolean` on a plugin.
static ConfigPropertyType	**BYTE_JAVA_LANG** ConfigPropertyType.BYTE_JAVA_LANG can be used to set a property of type `java.lang.Byte` on a plugin.
static ConfigPropertyType	**BYTE_PRIM** ConfigPropertyType.BYTE_PRIM can be used to set a property of type `byte` on a plugin.
static ConfigPropertyType	**CHAR_PRIM** ConfigPropertyType.CHAR_PRIM can be used to set a property of type `char` on a

[roperty Type](#)	plugin.
static [ConfigProperty Type](#)	**CHARACTER_JAVA_LANG** ConfigPropertyType.CHARACTER_JAVA_LANG can be used to set a property of type `java.lang.Character` on a plugin. **DOUBLE_JAVA_LANG** ConfigPropertyType.DOUBLE_JAVA_LANG can be used to set a property of type `java.lang.Double` on a plugin.
static [ConfigProp erty Type](#)	**DOUBLE_PRIM** ConfigPropertyType.DOUBLE_PRIM can be used to set a property of type `double` on a plugin.

[ertyType](#)	
staticConfigPropertyType	**FLOAT_JAVA_LANG** ConfigPropertyType.FLOAT_JAVA_LANG can be used to set a property of type `java.lang.Float` on a plugin.
staticConfigPropertyType	**FLOAT_PRIM** ConfigPropertyType.FLOAT_PRIM can be used to set a property of type `float` on a plugin.
staticConfigPropertyType	**INT_PRIM** ConfigPropertyType.INT_PRIM can be used to set a property of type `int` on a plugin.

[Type](#)	
static [ConfigPropertyType](#)	**INTEGER_JAVA_LANG** ConfigPropertyType.INTEGER_JAVA_LANG can be used to set a property of type `java.lang.Integer` on a plugin.
static [ConfigPropertyType](#)	**LONG_JAVA_LANG** ConfigPropertyType.LONG_JAVA_LANG can be used to set a property of type `java.lang.Long` on a plugin.
static [ConfigPropertyType](#)	**LONG_PRIM** ConfigPropertyType.LONG_PRIM can be used to set a property of type long on a plugin.

static ConfigPropertyType	**SHORT_JAVA_LANG** ConfigPropertyType.SHORT_JAVA_LANG can be used to set a property of type `java.lang.Short` on a plugin.
static ConfigPropertyType	**SHORT_PRIM** ConfigPropertyType.SHORT_PRIM can be used to set a property of type `short` on a plugin.
static ConfigPropertyType	**STRING_JAVA_LANG** ConfigPropertyType.STRING_JAVA_LANG can be used to set a property of type `java.lang.String` on a plugin.

Method Summary

boolean	**equals**(Object o)
int	**getId**() Get the raw value of this ConfigPropertyType.
int	**hashCode**()
String	**toString**()
static ConfigPropertyType	**valueOf**(byte id) Given the raw value of a ConfigPropertyType, this method returns a ConfigPropertyType object, or null if the raw value does not match an existing type.

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

INTEGER_JAVA_LANG

public static final ConfigPropertyType **INTEGER_JAVA_LANG**

ConfigPropertyType.INTEGER_JAVA_LANG can be used to set a property of type `java.lang.Integer` on a plugin.

INT_PRIM

public static final ConfigPropertyType **INT_PRIM**

ConfigPropertyType.INT_PRIM can be used to set a property of type `int` on a plugin.

BOOLEAN_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`BOOLEAN_JAVA_LANG`**

> ConfigPropertyType.BOOLEAN_JAVA_LANG can be used to set a property of type `java.lang.Boolean` on a plugin.

BOOLEAN_PRIM

`public static final` [ConfigPropertyType](#) **`BOOLEAN_PRIM`**

> ConfigPropertyType.BOOLEAN_PRIM can be used to set a property of type `boolean` on a plugin.

CHARACTER_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`CHARACTER_JAVA_LANG`**

> ConfigPropertyType.CHARACTER_JAVA_LANG can be used to set a property of type `java.lang.Character` on a plugin.

CHAR_PRIM

`public static final` [ConfigPropertyType](#) **`CHAR_PRIM`**

> ConfigPropertyType.CHAR_PRIM can be used to set a property of type `char` on a plugin.

BYTE_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`BYTE_JAVA_LANG`**

> ConfigPropertyType.BYTE_JAVA_LANG can be used to set a property of type `java.lang.Byte` on a plugin.

BYTE_PRIM

`public static final` [ConfigPropertyType](#) **`BYTE_PRIM`**

> ConfigPropertyType.BYTE_PRIM can be used to set a property of type `byte` on a plugin.

SHORT_JAVA_LANG

`public static final` [ConfigPropertyType](#) **`SHORT_JAVA_LANG`**

> ConfigPropertyType.SHORT_JAVA_LANG can be used to set a property of type `java.lang.Short` on a plugin.

SHORT_PRIM

`public static final` [ConfigPropertyType](#) **`SHORT_PRIM`**

> ConfigPropertyType.SHORT_PRIM can be used to set a property of type `short` on a plugin.

LONG_JAVA_LANG

`public static final ConfigPropertyType` **`LONG_JAVA_LANG`**

ConfigPropertyType.LONG_JAVA_LANG can be used to set a property of type `java.lang.Long` on a plugin.

LONG_PRIM

`public static final ConfigPropertyType` **`LONG_PRIM`**

ConfigPropertyType.LONG_PRIM can be used to set a property of type long on a plugin.

FLOAT_JAVA_LANG

`public static final ConfigPropertyType` **`FLOAT_JAVA_LANG`**

ConfigPropertyType.FLOAT_JAVA_LANG can be used to set a property of type `java.lang.Float` on a plugin.

FLOAT_PRIM

`public static final ConfigPropertyType` **`FLOAT_PRIM`**

ConfigPropertyType.FLOAT_PRIM can be used to set a property of type `float` on a plugin.

DOUBLE_JAVA_LANG

`public static final ConfigPropertyType` **`DOUBLE_JAVA_LANG`**

ConfigPropertyType.DOUBLE_JAVA_LANG can be used to set a property of type `java.lang.Double` on a plugin.

DOUBLE_PRIM

`public static final ConfigPropertyType` **`DOUBLE_PRIM`**

ConfigPropertyType.DOUBLE_PRIM can be used to set a property of type `double` on a plugin.

STRING_JAVA_LANG

`public static final ConfigPropertyType` **`STRING_JAVA_LANG`**

ConfigPropertyType.STRING_JAVA_LANG can be used to set a property of type `java.lang.String` on a plugin.

Method Detail

valueOf

`public static final ConfigPropertyType` **`valueOf`**`(byte id)`

Given the raw value of a ConfigPropertyType, this method returns a ConfigPropertyType object, or null if the raw value does not match an existing type. This method is used to

deserialize this object.

Parameters:
`id` - the raw value of a ConfigPropertyType
Returns:
the ConfigPropertyType corresponding to the raw input value
Since:
8.6, XC10 2.5

---

equals

`public boolean **equals**(Object o)`

Overrides:
`equals` in class `Object`

---

hashCode

`public int **hashCode**()`

Overrides:
`hashCode` in class `Object`

---

getId

`public int **getId**()`

Get the raw value of this ConfigPropertyType. This method is used to serialize this object.

Returns:
the raw value of this ConfigPropertyType.

---

toString

`public String **toString**()`

Overrides:
`toString` in class `Object`

---

com.ibm.websphere.objectgrid.config

Interface ConfigProperty

All Superinterfaces:

Serializable

```
public interface ConfigProperty
extends Serializable
```

ConfigProperty can be used to attach properties to a `Plugin`. A ConfigProperty has the following attributes:

- name: the property name
- value: the property value
- configPropertyType: the configuration property type

This ConfigProperty can be used to set the properties of a plugin. The name of the property should follow JaveBean convention. That is, for every property, there should be a corresponding set method in the plugin class.

Users can use `com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory.createConfigProperty(ConfigPropertyType, String, String)` to create a ConfigProperty object.

```
ConfigProperty evictorNameProp =
ObjectGridConfigFactory.createConfigProperty(ConfigProperty.STRING_JAVA_LANG, "evictorName",
"evictor1");
```
This creates a property "evictorName" with value "evictor1", and the type is `java.lang.String`. Use the `com.ibm.websphere.objectgrid.config.Plugin#addConfigProperty(ConfigProperty)` method to attach a ConfigProperty to a Plugin. When the Plugin is created, each ConfigProperty will have its corresponding set method called.

Continuing with the example above, attach the ConfigProperty to an Evictor Plugin.
`evictorPlugin.addConfigProperty(evictorNameProp);`

When this Evictor Plugin is created, the `setEvictorName(String) method will be called with the value "evictor1".`

Since:
WAS XD 6.0.1.2, XC10

Method Summary

ConfigPropertyType	**getConfigPropertyType**() Get the ConfigPropertyType of this object.

String	**getName**() Get the name of this object.
String	**getValue**() Get the value that is assigned to this ConfigProperty.
void	**setConfigPropertyType**(ConfigPropertyType configPropType) Set the ConfigPropertyType for this object.
void	**setName**(String name) Set the name of this object.
void	**setValue**(String value) Set the value of this ConfigProperty.

Method Detail

setConfigPropertyType

void **setConfigPropertyType**(ConfigPropertyType configPropType)

Set the ConfigPropertyType for this object. The Java primitives, their java.lang counterparts, and java.lang.String are the supported ConfigPropertyTypes.

Parameters:
configPropType -

See Also:
ConfigPropertyType.INTEGER_JAVA_LANG, ConfigPropertyType.INT_PRIM, ConfigPropertyType.BOOLEAN_JAVA_LANG, ConfigPropertyType.BOOLEAN_PRIM, ConfigPropertyType.CHARACTER_JAVA_LANG, ConfigPropertyType.CHAR_PRIM, ConfigPropertyType.BYTE_JAVA_LANG, ConfigPropertyType.BYTE_PRIM, ConfigPropertyType.SHORT_JAVA_LANG, ConfigPropertyType.SHORT_PRIM, ConfigPropertyType.LONG_JAVA_LANG, ConfigPropertyType.LONG_PRIM, ConfigPropertyType.FLOAT_JAVA_LANG, ConfigPropertyType.FLOAT_PRIM, ConfigPropertyType.DOUBLE_JAVA_LANG, ConfigPropertyType.DOUBLE_PRIM, ConfigPropertyType.STRING_JAVA_LANG

---

getConfigPropertyType

ConfigPropertyType **getConfigPropertyType**()

Get the ConfigPropertyType of this object.

Returns:
the ConfigPropertyType for this object

---

setValue

void **setValue**(String value)

Set the value of this ConfigProperty. This String value will be converted to the proper type, based on ConfigPropertyType assigned to this ConfigProperty

Parameters:
value - - will be converted to type and passed to the setter on the plugin

---

getValue

String **getValue**()

Get the value that is assigned to this ConfigProperty. This is the value that will be passed to the set method on the plugin.

Returns:
Returns the value.

---

setName

void **setName**(String name)

Set the name of this object. The Plugin that this ConfigProperty is attached to should have a setter that corresponds to this name. For example, if "size" is passed in as the name, then the Plugin must have a "setSize" method.

Parameters:
name - - name of the property

---

getName

String **getName**()

Get the name of this object. The name must have a corresponding set method on the Plugin.

Returns:
Returns the name.

---

com.ibm.websphere.objectgrid.config

Interface BackingMapConfiguration

public interface **BackingMapConfiguration**

A BackingMapConfiguration object can be used to override BackingMap settings on the client side. The `com.ibm.websphere.objectgrid.plugins.Evictor` and the `com.ibm.websphere.objectgrid.plugins.MapEventListener` Plugins can be overridden. Other Evictor related settings can be adjusted as well as client, near-cache specific options.

Use the `com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory.createBackingMapConfiguration(String)` method to create a BackingMapConfiguration

Since:

WAS XD 6.0.1.2, XC10

See Also:

Evictor, MapEventListener, Plugin, ObjectGridConfigFactory

Method Summary

void	**addPlugin**(Plugin plugin) Add a `Plugin` to this BackingMapConfiguration.
String	**getEvictionTriggers**() Gets the list of eviction triggers for this BackingMapConfiguration.
com.ibm.websphere.objectgrid.plugins.KeyOutputFormat	**getKeyOutputFormat**() Retrieves the data format for all data access APIs that return cache keys.

com.ibm.websphere.objectgrid.OutputFormat	
String	**getName**() Get the name of this BackingMapConfiguration
int	**getNumberOfBuckets**() **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the [isNearCacheEnabled()](#) flag to disable the near cache in the ObjectGrid configuration XML file.*
List	**getPlugins**() Get the Plugins that have been attached to this BackingMapConfiguration.
int	**getTimeToLive**() Gets the "time to live" for each map entry.
TTLType	**getTtlEvictorType**() Gets the "time to live" Evictor type for this BackingMapConfiguration.
com.ibm.websphere.objectgrid.OutputFormat	**getValueOutputFormat**() Retrieves the data format for all data access APIs that return cache values.

.OutputFormat	
Boolean	**isNearCacheEnabled**() If true, the client local cache is enabled for supported configurations.
Boolean	**isNearCacheInvalidationEnabled**() If true, clients with local caches are automatically invalidated when the data grid map is updated.
Boolean	**isNearCacheLastAccessTTLSyncEnabled**() If true, clients automatically send time-to-live access information to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.
void	**setEvictionTriggers**(String evictionTriggers) Sets the eviction triggers for this BackingMapConfiguration.
void	**setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat) Sets the data format for all data access APIs that return cache keys.
void	**setNearCacheEnabled**(Boolean nearCacheEnabled) Enables or disables the client local cache for supported configurations.
void	**setNearCacheInvalidationEnabled**(Boolean nearCacheInvalidationEnabled) Set to true to enable client near cache invalidation.
void	**setNearCacheLastAccessTTLSyncEnabled**(Boolean nearCacheLastAccessTTLSyncEnabled) Enables or disables time-to-live access information synchronization to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.
void	**setNumberOfBuckets**(int numBuckets) **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the isNearCacheEnabled() flag to disable the near cache in the ObjectGrid configuration XML file.*
void	**setPlugins**(List pluginList) Set the Plugins for this BackingMapConfiguration.

void	**setTimeToLive**(int seconds) Sets "time to live" of each BackingMap entry in seconds.
void	**setTtlEvictorType**(TTLType ttlEvictorType) Set the "time to live" Evictor type for this BackingMapConfiguration.
void	**setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat) Sets the data format for all data access APIs that return cache values.

Method Detail

getName

String **getName**()

Get the name of this BackingMapConfiguration

Returns:
The name of this BackingMapConfiguration

---

addPlugin

void **addPlugin**(Plugin plugin)

Add a `Plugin` to this BackingMapConfiguration. The Plugins that can be overridden on a client-side BackingMap are `com.ibm.websphere.objectgrid.plugins.Evictor` and `com.ibm.websphere.objectgrid.plugins.MapEventListener`.

Parameters:
`plugin` -
See Also:
setPlugins(List)

---

setPlugins

void **setPlugins**(List pluginList)

Set the Plugins for this BackingMapConfiguration. Any Plugins that were previously attached to this BackingMapConfiguration object will be overridden.

Parameters:
`pluginList` - - a List of Plugins
See Also:
addPlugin(Plugin)

---

getPlugins

List **getPlugins**()

Get the Plugins that have been attached to this BackingMapConfiguration.

Returns:

a List of `Plugin` objects

---

getNumberOfBuckets

`int` **`getNumberOfBuckets`**`()`

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*

Gets the number of buckets defined for this BackingMapConfiguration.

Returns:

the number of buckets defined

---

setNumberOfBuckets

`void` **`setNumberOfBuckets`**`(int numBuckets)`

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*

Sets the number of buckets for this BackingMapConfiguration. This will be used by the `BackingMap`.

The `BackingMap` implementation uses a hash map for its implementation. If there are a lot of entries in the `BackingMap` then more buckets means better performance because the risk of collisions is lower as the number of buckets grows. More buckets also means more concurrency.

Parameters:

`numBuckets` -

See Also:

`BackingMap.setNumberOfBuckets(int)`

---

getTimeToLive

`int` **`getTimeToLive`**`()`

Gets the "time to live" for each map entry. The value is in seconds.

Returns:

the "time to live" in seconds

---

setTimeToLive

`void` **`setTimeToLive`**`(int seconds)`

Sets "time to live" of each BackingMap entry in seconds.

If this method is not called, the lifetime of an entry is forever (or until the application explicitly removes or invalidates the entry, or a user defined `Evictor` evicts the entry).

Parameters:

`seconds` -

---

getTtlEvictorType

TTLType **getTtlEvictorType**()

Gets the "time to live" Evictor type for this BackingMapConfiguration. If setTtlEvictorType was not called, this method will return null and the BackingMap based off this BackingMapConfiguration will use `TTLType.NONE`

Returns:
the "time to live" Evictor type or null if `setTtlEvictorType(TTLType) was not called`
See Also:
setTimeToLive(int)

---

setTtlEvictorType

void **setTtlEvictorType**(TTLType ttlEvictorType)

Set the "time to live" Evictor type for this BackingMapConfiguration. This is used to determine how expiration time of a BackingMap entry is computed.

If this method is not called, TTLType.NONE is used to indicate the map entry has no expiration time (e.g. is allowed to live until explicitly removed or invalidated by the application, or evicted by a user defined Evictor).

Parameters:
ttlEvictorType -
See Also:
BackingMap.setTtlEvictorType(TTLType)

---

getEvictionTriggers

String **getEvictionTriggers**()

Gets the list of eviction triggers for this BackingMapConfiguration.

See BackingMap for a list of valid eviction triggers.

Returns:
a semicolon separated list of eviction triggers or null if setEvictionTriggers(String) was not called
Since:
WAS XD 6.1.0.3

---

setEvictionTriggers

void **setEvictionTriggers**(String evictionTriggers)

Sets the eviction triggers for this BackingMapConfiguration. All evictors will use the eviction supplied triggers.

See BackingMap for a list of valid eviction triggers.

Parameters:
evictionTriggers - a semicolon separated list of eviction triggers
Since:
WAS XD 6.1.0.3

---

isNearCacheInvalidationEnabled

Boolean **isNearCacheInvalidationEnabled**()

If true, clients with local caches are automatically invalidated when the data grid map is updated.

Returns:
true if client near cache invalidation is enabled, false if client near cache invalidation is disabled, or null if the override is not specified.
Since:
8.6, XC10 2.5

---

setNearCacheInvalidationEnabled

void **setNearCacheInvalidationEnabled**(Boolean nearCacheInvalidationEnabled)

Set to true to enable client near cache invalidation. When enabled, the client will receive events from the remote data grid to invalidate data from the local cache.

Parameters:
nearCacheInvalidationEnabled - If true, the client near cache invalidation is enabled. If false, invalidation is disabled. If null, the override is not specified and the client will use the setting from the remote data grid.
Since:
8.6, XC10 2.5

---

isNearCacheLastAccessTTLSyncEnabled

Boolean **isNearCacheLastAccessTTLSyncEnabled**()

If true, clients automatically send time-to-live access information to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.

Returns:
True if last-access time-to-live information is sent to the remote data grid, false if last-access time information is not sent, or null if the override is not specified.
Since:
8.6, XC10 2.5

---

setNearCacheLastAccessTTLSyncEnabled

void **setNearCacheLastAccessTTLSyncEnabled**(Boolean nearCacheLastAccessTTLSyncEnabled)

Enables or disables time-to-live access information synchronization to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.

Parameters:
nearCacheLastAccessTTLSyncEnabled - If true, the last-access time-to-live information is sent to the remote data grid. If false, the last-access information is not sent. If null, the override is not specified and the client will use the setting from the remote data grid.
Since:
8.6, XC10 2.5

---

isNearCacheEnabled

Boolean **isNearCacheEnabled**()

If true, the client local cache is enabled for supported configurations. The client near cache can only be enabled when using optimistic locking or when locking is disabled.

Returns:
True if the client near cache is enabled, false if the near cache is disabled, or null of the override is not specified.
Since:
8.6, XC10 2.5

---

setNearCacheEnabled

void **setNearCacheEnabled**(Boolean nearCacheEnabled)

Enables or disables the client local cache for supported configurations. The client near cache can only be enabled when using optimistic locking or when locking is disabled.

Parameters:
nearCacheEnabled - If true, the client local cache is enabled for supported configurations. If false, the client local cache is disabled. If null, the override is not specified and the client will use the setting from the remote data grid.

---

getKeyOutputFormat

com.ibm.websphere.objectgrid.OutputFormat **getKeyOutputFormat**()

Retrieves the data format for all data access APIs that return cache keys.

Returns:
the data format or null if the default should be used.
Since:
8.6, XC10 2.5

---

setKeyOutputFormat

void **setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat)

Sets the data format for all data access APIs that return cache keys.

Parameters:
dataFormat - the data format to use or null to use the default.
Since:
8.6, XC10 2.5

---

getValueOutputFormat

com.ibm.websphere.objectgrid.OutputFormat **getValueOutputFormat**()

Retrieves the data format for all data access APIs that return cache values.

Returns:
the data format.
Since:
8.6, XC10 2.5

---

setValueOutputFormat

void **setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat dataFormat)

Sets the data format for all data access APIs that return cache values.

Parameters:
dataFormat - the data format to use or null to use the default.
Since:
8.6, XC10 2.5

Package com.ibm.websphere.objectgrid.continuousquery

Interface Summary

ContinuousQueryCache<KeyType, ValueType>	A Continuous Query Cache contains the keys and optionally the values that match a defined continuous query.
ContinuousQueryFilter<KeyType,ValueType,AttributeType,MatchType>	An interface which provides an abstraction to check if an object matches the criteria defined in a filter.
ContinuousQueryListener<KeyType,ValueType>	This interface should be implemented to get a callback for a continuous query.
ContinuousQueryManager	The management interface used to define new continuous queries, retrieve an existing query, or destroy a query.
ContinuousQueryNotificationEvent<KeyType,ValueType>	An interface which defines notification metadata that is sent to continuous query listeners when the result set of a defined continous query changes.
ContinuousQueryTopic<KeyType,ValueType>	A Continuous Query Topic embodies the client side view of a defined Continuous Query.

Class Summary

ContinuousQueryManagerFactory	A factory class for obtaining the ContinuousQueryManager.

Enum Summary

ContinuousQueryNotificationEvent.Reason	Indicates the reason a notification event has occurred for this cache entry.

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryTopic<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried
`ValueType` - Type of the value object for the map being queried

`public interface` **`ContinuousQueryTopic<KeyType,ValueType>`**

A Continuous Query Topic embodies the client side view of a defined Continuous Query. The topic gives access to the client-side cached results of the continuous query and allows for management of Continuous Query Listeners for this topic.

Since:
8.6, XC10 2.5

Method Summary

void	**addListener**(ContinuousQueryListener<KeyType,ValueType> listener) Add a Continuous Query Listener to this topic.
Collection<ContinuousQueryListener<KeyType,ValueType>>	**getAllListeners**() Returns a shallow copy of the current set of listeners registered for this query topic.

[ContinuousQueryCache](#)<[KeyType](#), [ValueType](#)>	**[getCache](#)**() Return a continuous query cache object with methods to access the keys and values stored in the cache.
[String](#)	**[getName](#)**() Returns the generated unique name for this continuous query.
com.ibm.websphere	

e.objectgrid.OutputFormat	**getOutputFormat**() Return the format this query will use for keys and values.
List<Integer>	**getPartitions**() Return the list of partitions this query is defined against.
boolean	**isKeysOnlyCache**() Return true if the cache for this continuous query contains only keys.
boolean	**noCache**() Return true if the no caching option is enabled for this Continuous Query.
void	**removeAllListeners**() Removes all registered listeners from this query topic.
boolean	**removeListener**(ContinuousQueryListener<KeyType,ValueType> listener) Remove the listener from this continuous query's the set of listeners.

Method Detail

addListener

void **addListener**(ContinuousQueryListener<KeyType,ValueType> listener)

Add a Continuous Query Listener to this topic. The listener will be called when this topic receives updates to the query. There is no guarantee of the order that listeners will be called. Multiple adds of the same listener will result in only one call to the listener.

Parameters:

`listener` - the listener to add to this continuous query's set of listeners.

---

removeListener

boolean **removeListener**(ContinuousQueryListener<KeyType,ValueType> listener)

Remove the listener from this continuous query's the set of listeners. In flight notifications to this listener may continue after this method returns.

Parameters:

`listener` - - the continuous query listener to remove.

Returns:

true if the listener was removed.

---

getCache

ContinuousQueryCache<KeyType,ValueType> **getCache**()

Return a continuous query cache object with methods to access the keys and values stored in the cache.

Returns:

a Continuous Query Cache object for this topic.

---

getName

String **getName**()

Returns the generated unique name for this continuous query.

Returns:

a unique string.

---

isKeysOnlyCache

boolean **isKeysOnlyCache**()

Return true if the cache for this continuous query contains only keys. Return false if it also contains the value associated with the key.

Returns:

true if the cache does not contain values.

---

noCache

boolean **noCache**()

Return true if the no caching option is enabled for this Continuous Query.

Returns:

true if no keys or values are being cached.

getOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getOutputFormat`**`()`

Return the format this query will use for keys and values.

Returns:
the OutputFormat specified when this query was defined.

See Also:
`OutputFormat`

---

getAllListeners

`Collection<ContinuousQueryListener<KeyType,ValueType>>` **`getAllListeners`**`()`

Returns a shallow copy of the current set of listeners registered for this query topic.

Returns:
a new collection object containing all currently registered listeners.

---

removeAllListeners

`void` **`removeAllListeners`**`()`

Removes all registered listeners from this query topic. In flight notifications to listeners may continue after this method returns.

---

getPartitions

`List<Integer>` **`getPartitions`**`()`

Return the list of partitions this query is defined against. If the return value is null or an empty list, this query is defined against all partitions of the map.

Returns:
the array of partitions

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryNotificationEvent<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object in this event, from the key type used in the map being queried

`ValueType` - Type of the value object in this event, from the value type used in the map being queried

---

`public interface` **`ContinuousQueryNotificationEvent<KeyType,ValueType>`**

An interface which defines notification metadata that is sent to continuous query listeners when the result set of a defined continous query changes.

Since:

8.6, XC10 2.5

Nested Class Summary

static class	**ContinuousQueryNotificationEvent.Reason** Indicates the reason a notification event has occurred for this cache entry.

Method Summary

KeyType	**getKey**() Retrieve the key of the cache entry that triggered this notification.
ContinuousQu	

[EntryNotificationEvent.Reason](#)	**getReason**() Retrieve the reason for this notification event
com.ibm.websphere.objectgrid.SessionHandle	**getSessionHandle**() Retrieve a SessionHandle object which can be used to retrieve the cache entry that triggered this notification.
[ValueType](#)	**getValue**() Retrieve the value of the cache entry that triggered this notification, if the query was

[V] [p] [e]	configured to return the value object for matches.
boolean	**[isKeysOnly]**() If true, this notification only includes the key of the matching cache entry.

Method Detail

getKey

[KeyType] **getKey**()

Retrieve the key of the cache entry that triggered this notification. This object is a reference to the object stored in the [ContinuousQueryCache], if the cache has been enabled. Therefore it should not be modified.

Returns:
the cache entry key

---

getValue

[ValueType] **getValue**()

Retrieve the value of the cache entry that triggered this notification, if the query was configured to return the value object for matches. This object is a reference to the object stored in the [ContinuousQueryCache], if the cache has been enabled. Therefore it should not be modified.

Returns:
the cache entry value

---

getSessionHandle

com.ibm.websphere.objectgrid.SessionHandle **getSessionHandle**()

Retrieve a SessionHandle object which can be used to retrieve the cache entry that triggered this notification. This method can only be invoked when using a PER_CONTAINER placement strategy for the map.

Returns:
the SessionHandle for the partition that owns the cache entry.
See Also:
MapSet.getPlacementStrategy(), [Session.setSessionHandle(SessionHandle)]

---

getReason

[ContinuousQueryNotificationEvent.Reason] **getReason**()

Retrieve the reason for this notification event

Returns:

the Reason enum

---

isKeysOnly

`boolean **isKeysOnly**()`

If true, this notification only includes the key of the matching cache entry. If false, this notification also includes the value

Returns:
whether or not this notification includes the cache value.
See Also:
getValue()

---

com.ibm.websphere.objectgrid.continuousquery

Enum ContinuousQueryNotificationEvent.Reason

java.lang.Object

java.lang.Enum<ContinuousQueryNotificationEvent.Reason>

com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryNotificationEvent.Reason

All Implemented Interfaces:

Serializable, Comparable<ContinuousQueryNotificationEvent.Reason>

Enclosing interface:

ContinuousQueryNotificationEvent<KeyType,ValueType>

```
public static enum ContinuousQueryNotificationEvent.Reason
extends Enum<ContinuousQueryNotificationEvent.Reason>
```

Indicates the reason a notification event has occurred for this cache entry.

Enum Constant Summary

ADDED The cache entry was inserted into the grid and matches the query, or an entry which did not match the query was updated such that it now matches the query.
CLEAR A cache entry which matched the query was removed from the grid due to a map clear operation.
REMOVED The cache entry, which previously matched the query, was removed from the grid or updated such that it no longer matches.
UPDATED The cache entry, which previously matched the query, was updated and still matches the query.

Method Summary

static Continuous	

QueryNotificationEvent.Reason	**valueOf**(String name) Returns the enum constant of this type with the specified name.
static QueryNotificationEvent.Reason[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

ADDED

```
public static final ContinuousQueryNotificationEvent.Reason ADDED
```

The cache entry was inserted into the grid and matches the query, or an entry which did not match the query was updated such that it now matches the query.

REMOVED

```
public static final ContinuousQueryNotificationEvent.Reason REMOVED
```

The cache entry, which previously matched the query, was removed from the grid or updated such that it no longer matches.

UPDATED

```
public static final ContinuousQueryNotificationEvent.Reason UPDATED
```

The cache entry, which previously matched the query, was updated and still matches the query. Note: The result of getValue() for REMOVE events will always be null, even if the query was defined to include values.

See Also:

ContinuousQueryManager.defineContinuousQuery(java.lang.String, com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryFilter, boolean, boolean, boolean, java.util.Collection>, boolean, com.ibm.websphere.objectgrid.OutputFormat, java.util.List)

CLEAR

```
public static final ContinuousQueryNotificationEvent.Reason CLEAR
```

A cache entry which matched the query was removed from the grid due to a map clear operation.

Method Detail

values

```
public static ContinuousQueryNotificationEvent.Reason[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (ContinuousQueryNotificationEvent.Reason c : ContinuousQueryNotificationEvent.Reason.valu
es())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

---

valueOf

`public static ContinuousQueryNotificationEvent.Reason **valueOf**(String name)`

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`IllegalArgumentException` - if this enum type has no constant with the specified name

`NullPointerException` - if the argument is null

---

com.ibm.websphere.objectgrid.continuousquery

Class ContinuousQueryManagerFactory

java.lang.Object

com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryManagerFactory

---

```
public final class ContinuousQueryManagerFactory
extends Object
```

A factory class for obtaining the ContinuousQueryManager.

Since:

8.6, XC10 2.5

Constructor Summary

ContinuousQueryManagerFactory()

Method Summary

static ContinuousQueryManager	**getManager**(ObjectGrid objGrid) Factory that returns the single instance of a ContinuousQueryManager for the given object grid.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ContinuousQueryManagerFactory

public **ContinuousQueryManagerFactory**()

Method Detail

getManager

public static ContinuousQueryManager **getManager**(ObjectGrid objGrid)

Factory that returns the single instance of a ContinuousQueryManager for the given object grid.

Parameters:

`objGrid` - - the ObjectGrid this Continuous Query Manager will define continuous queries for.

Returns:

the ContinuousQueryManager instance.

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryManager

public interface **ContinuousQueryManager**

The management interface used to define new continuous queries, retrieve an existing query, or destroy a query. An instance can be retrieved via ContinuousQueryManagerFactory.getManager(com.ibm.websphere.objectgrid.ObjectGrid)

Since:

8.6, XC10 2.5

Method Summary

<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType>	**defineContinuousQuery**(String mapName, ContinuousQueryFilter filter, boolean keysOnly, boolean returnInitialResultSet, boolean notifyOnUpdated) Define a new continuous query.
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType>	**defineContinuousQuery**(String mapName, ContinuousQueryFilter filter, boolean keysOnly, boolean returnInitialResultSet, boolean notifyOnUpdated, Collection<ContinuousQueryListener<KeyType,ValueType>> continuousQueryListeners, boolean noCache, com.ibm.websphere.objectgrid.OutputFormat format, List<Integer> partitionSubset) Define a new continuous query.
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType>	**defineContinuousQuery**(String mapName, ContinuousQueryFilter filter, boolean keysOnly, boolean returnInitialResultSet, boolean notifyOnUpdated, List<Integer> partitionSubset) Define a new continuous query.
List<ContinuousQueryTopic<?,?>>	**getDefinedContinuousQueries**() Returns a list view of the currently defined Continuous Query Topics.
boolean	**removeContinuousQuery**(ContinuousQueryTopic<?,?> topic) Remove a defined continuous query.

Method Detail

defineContinuousQuery

<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType> **defineContinuousQuery**(String mapName,
ContinuousQueryFilter filter,
boolean keysOnly,
boolean returnInitialResultSet,
boolean notifyOn

```
Updated,
                                                                    Collection<Conti
nuousQueryListener<KeyType,ValueType>> continuousQueryListeners,
                                                                    boolean noCache,
                                                                    com.ibm.webspher
e.objectgrid.OutputFormat format,
                                                                    List<Integer> pa
rtitionSubset)
                                                        throws ContinuousQueryIncompatibleDu
plicateException,
                                                               UndefinedMapException
```

Define a new continuous query. This query will run in the grid container processes. When cache entries that match the filter criteria are inserted or deleted, the client's `ContinuousQueryCache` will be updated and any `ContinuousQueryListener` implementations will be invoked. This will also occur if a cache entry is updated such that it now matches, or no longer matches, the filter criteria. Optionally, notifications can be sent for update operations to cache entries that previously matched the criteria, and still match the criteria after the update.

Parameters:

`mapName` - - the name of the map the continuous query will be defined on.
`filter` - - the filter that the will determine the contents of the continuous query. Custom filter logic can be implemented by extending `AbstractCQFilter`.
`keysOnly` - - true if the query should only return the keys of items that satisfy the query; false to also send the values as well as the keys. If false, the `ContinuousQueryCache` and `ContinuousQueryNotificationEvent` objects will include the value associated with cache entries that match the query.
`returnInitialResultSet` - - true to run the filter on all existing matches in the map at the time of the query definition and return the items to the cache. If false, the client will only be notified of matches which occur after the query is defined. Returning the initial result set will have a high computational requirement for large maps.
`notifyOnUpdated` - - true causes this topic to get called every time an existing item that satisfies this query gets updated and continues to satisfy this query. This will drive `ContinuousQueryListener` call backs as well as updated values in the query cache if keysOnly is false.
`continuousQueryListeners` - - used to register listeners before the query is defined. These listeners will be guaranteed to be invoked for any entries in the initial result set, as well as changes to the result set which might occur before `ContinuousQueryTopic.addListener(ContinuousQueryListener)` can be invoked. Can be null.
`noCache` - - true to indicate that no caching of keys or values should occur on the client. Continuous Query listeners will still be notified.
`format` - - if `OutputFormat.RAW` is specified, keys and values returned from the `ContinuousQueryCache` and `ContinuousQueryListener` will be in `SerializedEntry` format when a `DataSerializer` is defined on the map. - if `OutputFormat.NATIVE` is specified, or when no `DataSerializer` is defined, keys and values will be returned as Objects.
`partitionSubset` - - Used to indicate that this continuous query should only be defined on the specified subset of partitions. Passing null or an empty list indicates this continuous query should be defined on all existing and future partitions containing this map. An IllegalArgumentException may be thrown if an invalid partition ID is passed.

Returns:

The `ContinuousQueryTopic` for this continuous query. If the call to this method would result in an topic which is identical to an existing topic being created, the existing instance is returned.

Throws:

`ContinuousQueryIncompatibleDuplicateException`
`UndefinedMapException`
`IllegalArgumentException`

---

defineContinuousQuery

```
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType> defineContinuousQuery(String mapName,
                                                                                   ContinuousQueryF
ilter filter,
                                                                                   boolean keysOnly
,
                                                                                   boolean returnIn
itialResultSet,
                                                                                   boolean notifyOn
Updated,
                                                                                   List<Integer> pa
rtitionSubset)
                                                                            throws ContinuousQueryIncompatibleDu
plicateException,
                                                                                   UndefinedMapException
```

Define a new continuous query. This query will run in the grid container processes. When cache entries that match the filter criteria are inserted or deleted, the client's `ContinuousQueryCache` will be updated and any `ContinuousQueryListener` implementations will be invoked. This will also occur if a cache entry is updated such that it now matches, or no longer matches, the filter criteria. Optionally, notifications can be sent for update operations to cache entries that previously matched the criteria, and still match the criteria after the update.

Parameters:

`mapName` - - the name of the map the continuous query will be defined on.
`filter` - - the filter that the will determine the contents of the continuous query. Custom filter logic can be implemented by extending `AbstractCQFilter`.
`keysOnly` - - true if the query should only return the keys of items that satisfy the query; false to also send the values as well as the keys. If false, the `ContinuousQueryCache` and `ContinuousQueryNotificationEvent` objects will include the value associated with cache entries that match the query.
`returnInitialResultSet` - - true to run the filter on all existing matches in the map at the time of the query definition and return the items to the cache. If false, the client will only be notified of matches which occur after the query is defined. Returning the initial result set will have a high computational requirement for large maps.
`notifyOnUpdated` - - true causes this topic to get called every time an existing item that satisfies this query gets updated and continues to satisfy this query. This will drive `ContinuousQueryListener` call backs as well as updated values in the query cache if keysOnly is false.
`partitionSubset` - - Used to indicate that this continuous query should only be defined on the specified subset of partitions. Passing null or an empty list indicates this continuous query should be defined on all existing and future partitions containing this map. An IllegalArgumentException may be thrown if an invalid partition ID is passed.

Returns:

The `ContinuousQueryTopic` for this continuous query. If the call to this method would result in an topic which is identical to an existing topic being created, the existing instance is returned.

Throws:

`ContinuousQueryIncompatibleDuplicateException`
`UndefinedMapException`
`IllegalArgumentException`

---

defineContinuousQuery

```
<KeyType,ValueType> ContinuousQueryTopic<KeyType,ValueType> defineContinuousQuery(String mapName,
                                                                                   ContinuousQueryF
ilter filter,
                                                                                   boolean keysOnly
,
                                                                                   boolean returnIn
itialResultSet,
                                                                                   boolean notifyOn
```

Updated)
throws ContinuousQueryIncompatibleDuplicateException,
UndefinedMapException

Define a new continuous query. This query will run in the grid container processes. When cache entries that match the filter criteria are inserted or deleted, the client's ContinuousQueryCache will be updated and any ContinuousQueryListener implementations will be invoked. This will also occur if a cache entry is updated such that it now matches, or no longer matches, the filter criteria. Optionally, notifications can be sent for update operations to cache entries that previously matched the criteria, and still match the criteria after the update. This query will be applied to all partitions.

Parameters:

`mapName` - - the name of the map the continuous query will be defined on.
`filter` - - the filter that the will determine the contents of the continuous query. Custom filter logic can be implemented by extending AbstractCQFilter.
`keysOnly` - - true if the query should only return the keys of items that satisfy the query; false to also send the values as well as the keys. If false, the ContinuousQueryCache and ContinuousQueryNotificationEvent objects will include the value associated with cache entries that match the query.
`returnInitialResultSet` - - true to run the filter on all existing matches in the map at the time of the query definition and return the items to the cache. If false, the client will only be notified of matches which occur after the query is defined. Returning the initial result set will have a high computational requirement for large maps.
`notifyOnUpdated` - - true causes this topic to get called every time an existing item that satisfies this query gets updated and continues to satisfy this query. This will drive ContinuousQueryListener call backs as well as updated values in the query cache if keysOnly is false.

Returns:

The ContinuousQueryTopic for this continuous query. If the call to this method would result in an topic which is identical to an existing topic being created, the existing instance is returned.

Throws:

ContinuousQueryIncompatibleDuplicateException
UndefinedMapException
IllegalArgumentException

---

removeContinuousQuery

boolean **removeContinuousQuery**(ContinuousQueryTopic<?,?> topic)

Remove a defined continuous query. Does the opposite of defineContinuousQuery(String, ContinuousQueryFilter, boolean, boolean, boolean). If the number of calls to this method for a given topic equals the number of calls to defineContinuousQuery that return the specified topic, the topic will cease receiving updates from the server, all listeners of this topic will be removed and the reference to this topic will be invalid.

Parameters:

`topic` - - the topic to remove

Returns:

true if the topic was removed, false otherwise

Throws:

IllegalArgumentException

---

getDefinedContinuousQueries

List<ContinuousQueryTopic<?,?>> **getDefinedContinuousQueries**()

Returns a list view of the currently defined Continuous Query Topics.

Returns:

a list of Continuous Query Topics

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryListener<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried

`ValueType` - Type of the value object for the map being queried

`public interface` **`ContinuousQueryListener<KeyType,ValueType>`**

This interface should be implemented to get a callback for a continuous query. When a Continuous Query listener is added to a Continuous Query Topic, that listener will be called after every change to the current set of results matching the continuous query.

Since:

8.6, XC10 2.5

Method Summary

void	**cacheUpdated**(ContinuousQueryNotificationEvent<KeyType,ValueType> event) This method will be called by every ContinuousQueryTopic that this listener is added to.

Method Detail

cacheUpdated

`void` **`cacheUpdated`**(ContinuousQueryNotificationEvent<KeyType,ValueType> event)

This method will be called by every ContinuousQueryTopic that this listener is added to. The caller will capture any Throwables thrown from calls to this method and create an FFDC entry. This ContinuousQueryListener will not be called again for the same ContinuousQueryNotificationEvent. Additionally, the performance of this ContinuousQueryListener may make an impact on the speed with which other ContinuousQueryListeners are called.

Parameters:

`event` - - ContinuousQueryNotificationEvent containing information about the change in the query result set

See Also:

ContinuousQueryTopic.addListener(ContinuousQueryListener), ContinuousQueryNotificationEvent

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryFilter<KeyType,ValueType,AttributeType,MatchType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried
`ValueType` - Type of the value object for the map being queried
`AttributeType` - Type of the attribute referenced by the attribute path
`MatchType` - Type of the object being compared to

All Superinterfaces:

Serializable

All Known Implementing Classes:

AbstractCQFilter, AndFilter, BinaryLogicalFilter, CompareFilter, EQFilter, FalseFilter, GTEFilter, GTFilter, IsNotNullFilter, IsNullFilter, LTEFilter, LTFilter, MatchFilter, NEQFilter, NotFilter, NotMatchFilter, OrFilter, TrueFilter

---

```
public interface ContinuousQueryFilter<KeyType,ValueType,AttributeType,MatchType>
extends Serializable
```

An interface which provides an abstraction to check if an object matches the criteria defined in a filter. Filters are invoked within the eXtreme Scale containers hosting partitions for the map on which the query is defined. All implementations must extend `AbstractCQFilter`.

Since:

8.6, XC10 2.5

Field Summary

static String	**POJO_ADDRESSABLEKEYNAME** The name used to identify the key objects in POJO maps.
static String	**POJO_PATHSEPARATOR** The path separator to use when identifying attribute paths for POJO maps.

boolean	**filter**(FilterContent<KeyType,ValueType> content) Checks if the supplied object passes the filter.

Field Detail

POJO_ADDRESSABLEKEYNAME

static final String **POJO_ADDRESSABLEKEYNAME**

The name used to identify the key objects in POJO maps. Not applicable to maps using XDF or a custom MapSerializedPlugin.

See Also:
CompareFilter.CompareFilter(String, Object), Constant Field Values

---

POJO_PATHSEPARATOR

static final String **POJO_PATHSEPARATOR**

The path separator to use when identifying attribute paths for POJO maps. Not applicable to maps using XDF or a custom MapSerializedPlugin.

See Also:
CompareFilter.CompareFilter(String, Object), Constant Field Values

Method Detail

filter

boolean **filter**(FilterContent<KeyType,ValueType> content)
throws ContinuousQueryException

Checks if the supplied object passes the filter.

Parameters:
content - A representation of the cache entry to be checked
Returns:
true if the object matches the filtering criteria, false otherwise
Throws:
ContinuousQueryException

---

com.ibm.websphere.objectgrid.continuousquery

Interface ContinuousQueryCache<KeyType,ValueType>

Type Parameters:

`KeyType` - Type of the key object for the map being queried
`ValueType` - Type of the value object for the map being queried

`public interface` **`ContinuousQueryCache<KeyType,ValueType>`**

A Continuous Query Cache contains the keys and optionally the values that match a defined continuous query. The contents of this cache arrive asynchronously from the grid for which the query is defined. If the query is defined such that only the keys are stored in the continuous query cache then all operations that return a value will return null.

Since:
8.6, XC10 2.5

Method Summary

boolean	**containsKey**(KeyType key) Returns true if the cache contains the given key.
boolean	**containsValue**(ValueType value) Returns true if the cache contains the given value.
ValueType	**get**(KeyType key) Returns the value for the given key.
List<com.ibm.	

org.websphere.objectgrid.SessionHandle>	**getSessionHandles**(KeyType key) Returns SessionHandle objects that can be used to retrieve all query matches associated with this key.
boolean	**includesValues**() Returns true if this cache includes the values associated with the keys that have matched the query.
Set<KeyType>	**keySnapshot**() Returns a Set view of a snapshot of the keys contained in the cache.
int	**size**() Return the number of keys in the cache.

Method Detail

keySnapshot

Set<KeyType> **keySnapshot**()

Returns a Set view of a snapshot of the keys contained in the cache. The objects in the set are references to the keys in the query cache, therefore they should not be modified.

Returns:

a set view of the keys in the cache.

size

`int` **`size`**`()`

Return the number of keys in the cache.

Returns:
the number of keys in the cache.

---

get

`ValueType` **`get`**`(KeyType key)`

Returns the value for the given key. The object returned is a reference to the value in the query cache, therefore it should not be modified.

Parameters:
`key` - - the key whose associated value is to be returned
Returns:
the value in the cache for the specified key or null if either the cache does not contain this key or if the cache is configured to not contain values.

---

getSessionHandles

`List<com.ibm.websphere.objectgrid.SessionHandle>` **`getSessionHandles`**`(KeyType key)`

Returns `SessionHandle` objects that can be used to retrieve all query matches associated with this key. This method can only be invoked when using a PER_CONTAINER placement strategy for the map.

Parameters:
`key` - - the key for which SessionHandles will be returned
Returns:
a List of SessionHandles, one for each instance of this key in the grid.
See Also:
`MapSet.getPlacementStrategy()`

---

containsKey

`boolean` **`containsKey`**`(KeyType key)`

Returns true if the cache contains the given key.

Parameters:
`key` - - key to check for
Returns:
true if the specified key is in the cache at the time of the request.

---

containsValue

`boolean` **`containsValue`**`(ValueType value)`

Returns true if the cache contains the given value.

Parameters:
`value` - - value to check for
Returns:

true if the specified object is in the cache at the time of the request.

---

includesValues

`boolean **includesValues**()`

Returns true if this cache includes the values associated with the keys that have matched the query. If true, `containsValue(Object)` and `get(Object)` will return usable values, otherwise they will return null.

Returns:
true if this query cache contains values in addition to keys.

See Also:
`ContinuousQueryManager.defineContinuousQuery(java.lang.String, com.ibm.websphere.objectgrid.continuousquery.ContinuousQueryFilter, boolean, boolean, boolean, java.util.Collection>, boolean, com.ibm.websphere.objectgrid.OutputFormat, java.util.List)`

Package com.ibm.websphere.objectgrid.management

This package contains the interfaces for all ObjectGrid MBeans.

See:

Description

Interface Summary

AgentManagerMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific Agent on a server process.
CatalogServiceManagementMBean	This MBean interface allows user to manipulate the behaviors of heartbeat and leader manager and other catalog service specific actions.
ContainerMBean	This MBean interface allows a client process to perform operations on and get status from an ObjectGrid container running in a dynamic environment.
DynamicServerMBean	This MBean interface allows a client process to access different attributes about a specific server process in a dynamic environment.
HashIndexMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific HashIndex on a server process.
MapMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific map on a server process.
ObjectGridMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific ObjectGrid on a server process.
PlacementMediationServiceMBean	This MBean interface allows a client process to perform operations on and get status from the PlacementMediationService running in a dynamic environment.
PlacementServiceMBean	This MBean interface allows a client process to perform operations on and get status from the PlacementService running in a dynamic environment.
QueryManagerMBean	This MBean interface allows a client process to perform operations on and get status from an ObjectGrid Query Manager running in a dynamic environment.
QuorumManagerMBean	Each catalog service has a QuorumManager and an associated MBean.
ServerMBean	This MBean interface allows a client process to access different attributes about a specific server process.
SessionMBean	This MBean interface allows a client process to access different attributes and statistical data about a specific session.
ShardMBean	This MBean interface allows a client process to perform operations on and get status from a shard running in a dynamic environment.
ThreadPoolMBean	This MBean interface allows user to access the thread pool properties.

Package com.ibm.websphere.objectgrid.management Description

This package contains the interfaces for all ObjectGrid MBeans.

Overview

Each MBean interface has several methods to administer and monitor ObjectGrid services and components.

---

com.ibm.websphere.objectgrid.management

Interface ThreadPoolMBean

public interface **ThreadPoolMBean**

This MBean interface allows user to access the thread pool properties. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=ThreadPool
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
7.0.0.0 FIX2, XC10

Method Summary

int	**getActiveThreadCount**() Retrieves the approximate number of active threads in the pool.
int	**getMaximumSize**() Retrieves the maximum number of threads in the pool.
int	**getMinimumSize**() Retrieves the minimum number of threads in the pool.
String	**getName**() Retrieves the name of the ThreadPool.
void	**setMaximumSize**(int size) Sets the maximum thread pool size.
void	**setMinimumSize**(int size) Sets the minimum thread pool size.

Method Detail

setMaximumSize

`void` **`setMaximumSize`**`(int size)`

Sets the maximum thread pool size.

Parameters:
`size` - the maximum number of threads.

---

getMaximumSize

`int` **`getMaximumSize`**`()`

Retrieves the maximum number of threads in the pool.

Returns:
the maximum number of threads in the pool

---

setMinimumSize

`void` **`setMinimumSize`**`(int size)`

Sets the minimum thread pool size.

Parameters:
`size` - the minimum number of threads.

---

getMinimumSize

`int` **`getMinimumSize`**`()`

Retrieves the minimum number of threads in the pool.

Returns:
the minimum number of threads in the pool

---

getActiveThreadCount

`int` **`getActiveThreadCount`**`()`

Retrieves the approximate number of active threads in the pool.

Returns:
the number of active threads in the pool in use

---

getName

`String` **`getName`**`()`

Retrieves the name of the ThreadPool.

Returns:
the name of the ThreadPool

---

com.ibm.websphere.objectgrid.management

Interface ShardMBean

public interface **ShardMBean**

This MBean interface allows a client process to perform operations on and get status from a shard running in a dynamic environment. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=Shard,name=<objectgrid>,objectgrid=<objectgrid>,mapset=<mapset>
,partition=<partition id>,container=<container>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static String	**ROLE_SWAP_REQUESTED_WITH_SAME_TYPE** Indicates that this shard is the same type of shard as the requested swap type.
static String	**ROLE_SWAP_SUCCESSFUL** Indicates that the role swap was executed successfully.
static String	**ROLE_SWAP_TIMEOUT** Indicates that this shard has timed out waiting to inherit its requested role
st	

static String	**TYPE_INACTIVE** Indicates the shard type is the inactive role.
static String	**TYPE_PRIMARY** Indicates the shard type is the primary role.
static String	**TYPE_REPLICA_ASYNCHRONOUS** Indicates the shard type is the asynchronous replica role.
static String	**TYPE_REPLICA_SYNCHRONOUS** Indicates the shard type is the synchronous replica role.

Method Summary

long	**getActiveRequestCount**() Retrieves the number of requests currently being processed by this shard.
String	**getContainerName**() Retrieves the name of the container that is hosting this shard.
String	**getDomainName**() Retrieve the name of the catalog server grouping administering this shard.
long	**getForwardedRequestCount**() Retrieves the number of requests that this shard has forwarded since its inception.

String	**getMapSetName**() Retrieve the name of the MapSet in which the shard resides.
String	**getObjectGridName**() Retrieve the name of the ObjectGrid in which the shard resides.
String	**getPartitionName**() Retreive the name of the partition in which the shard resides.
long	**getProcessedRequestCount**() Retrieves the number of requests that this shard has processed since its inception.
String	**getState**() Retrieve the state of the shard.
long	**getTotalRequestCount**() Retrieves the number of requests that this shard has processed or forwarded since its inception.
String	**getType**() Retrieve the type of the shard.
String	**swapWithPrimary**() Causes this shard to swap roles with the primary shard for the partition.

Field Detail

ROLE_SWAP_SUCCESSFUL

```
static final String ROLE_SWAP_SUCCESSFUL
```

Indicates that the role swap was executed successfully.

Since:
7.1.0.0 FIX1

See Also:
swapWithPrimary(), Constant Field Values

ROLE_SWAP_REQUESTED_WITH_SAME_TYPE

`static final String` **`ROLE_SWAP_REQUESTED_WITH_SAME_TYPE`**

Indicates that this shard is the same type of shard as the requested swap type. No swap will be executed.

Since:
7.1.0.0 FIX1

See Also:
`swapWithPrimary()`, Constant Field Values

ROLE_SWAP_TIMEOUT

`static final String` **`ROLE_SWAP_TIMEOUT`**

Indicates that this shard has timed out waiting to inherit its requested role

Since:
7.1.0.0 FIX1

See Also:
`swapWithPrimary()`, Constant Field Values

TYPE_PRIMARY

`static final String` **`TYPE_PRIMARY`**

Indicates the shard type is the primary role. This means that this is the shard that handles all updates and coordinates state transitions with the replicas.

See Also:
Constant Field Values

TYPE_REPLICA_SYNCHRONOUS

`static final String` **`TYPE_REPLICA_SYNCHRONOUS`**

Indicates the shard type is the synchronous replica role. This means that this shard is receiving state updates from another shard that is acting as primary.

See Also:
Constant Field Values

TYPE_REPLICA_ASYNCHRONOUS

`static final String` **`TYPE_REPLICA_ASYNCHRONOUS`**

Indicates the shard type is the asynchronous replica role. This means that this shard is receiving state updates from another shard that is acting as primary.

See Also:
Constant Field Values

TYPE_INACTIVE

`static final String` **`TYPE_INACTIVE`**

Indicates the shard type is the inactive role. This means that this shard is not actively enrolled in the partition.

See Also:
Constant Field Values

Method Detail

getObjectGridName

`String` **`getObjectGridName`**`()`

Retrieve the name of the ObjectGrid in which the shard resides.

Returns:
The ObjectGrid name.

---

getMapSetName

`String` **`getMapSetName`**`()`

Retrieve the name of the MapSet in which the shard resides.

Returns:
The MapSet name.

---

getPartitionName

`String` **`getPartitionName`**`()`

Retreive the name of the partition in which the shard resides.

Returns:
The partition name.

---

getType

`String` **`getType`**`()`

Retrieve the type of the shard.

Returns:
The shard type.

---

getDomainName

`String` **`getDomainName`**`()`

Retrieve the name of the catalog server grouping administering this shard.

Returns:
The domain name.

---

getState

`String` **`getState`**`()`

Retrieve the state of the shard.

Returns:
The shard state.

---

getTotalRequestCount

`long` **`getTotalRequestCount`**`()`

Retrieves the number of requests that this shard has processed or forwarded since its inception.

Returns:
A count of the total number of requests.

---

getActiveRequestCount

`long` **`getActiveRequestCount`**`()`

Retrieves the number of requests currently being processed by this shard.

Returns:
A count of the active requests.

---

getForwardedRequestCount

`long` **`getForwardedRequestCount`**`()`

Retrieves the number of requests that this shard has forwarded since its inception.

Returns:
A count of the total number of forwarded requests.

---

getProcessedRequestCount

`long` **`getProcessedRequestCount`**`()`

Retrieves the number of requests that this shard has processed since its inception.

Returns:
A count of the total number of processed requests.

---

getContainerName

`String` **`getContainerName`**`()`

Retrieves the name of the container that is hosting this shard.

Returns:
The name of the container.
Since:
WAS XD 6.1.0.3

---

swapWithPrimary

`String` **`swapWithPrimary`**`()`

Causes this shard to swap roles with the primary shard for the partition. This shard becomes the primary while the shard that was previously the primary inherits this shard's former role.

If the role swap is not complete within 10 seconds, this operation will timeout.

Returns:

String the contains the return code of the operation

Since:

7.1.0.0 FIX1

See Also:

ROLE_SWAP_SUCCESSFUL, ROLE_SWAP_REQUESTED_WITH_SAME_TYPE, ROLE_SWAP_TIMEOUT

---

com.ibm.websphere.objectgrid.management

Interface SessionMBean

public interface **SessionMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific session. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=Session,name=<id>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
WAS XD 6.0.1, XC10

Method Summary

long	**getAccessedSessionsCount**() Gets acessed sessions count
long	**getAccessToNonExistentSessionCount**() Gets access to non-existent session count
long	**getActiveSessionsCount**() Gets active sessions count
long	**getAffinityBreaksCount**() Gets affinity breaks count
long	**getCacheDiscardsCount**() Gets cache discards count
long	**getCreatedSessionsCount**() Gets the map count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getInvalidatedByTimeoutCount**() Gets invalidated by timeout count
l	

long	**getInvalidatedSessionsCount**() Gets invalidated sessions count
long	**getMemoryCount**() Gets memory count
String	**getSessionID**() Gets the ID of the session instance associated with this MBean.
String	**getSessionStatsModule**() Gets a string representation of the `SessionStatsModule attributes loaded up by the retrieveStatsModule() method.`
com.ibm.websphere.objectgrid.stats.SessionStatsModule	**retrieveStatsModule**() Gets the `SessionStatsModule` used to retrieve statistics associated with the session for this MBean.

e	

Method Detail

retrieveStatsModule

`com.ibm.websphere.objectgrid.stats.SessionStatsModule` **`retrieveStatsModule`**`()`

Gets the `SessionStatsModule` used to retrieve statistics associated with the session for this MBean.

Returns:
an SessionStatsModule for statistics associated with this session

See Also:
`SessionStatsModule`

---

getSessionID

`String` **`getSessionID`**`()`

Gets the ID of the session instance associated with this MBean.

Returns:
the ID of the session instance associated with this MBean.

---

getSessionStatsModule

`String` **`getSessionStatsModule`**`()`

Gets a string representation of the `SessionStatsModule` attributes loaded up by the `retrieveStatsModule()` method.

Returns:
String form of SessionStatsModule

See Also:
`retrieveStatsModule()`, SessionStatsModule

---

getCreatedSessionsCount

`long` **`getCreatedSessionsCount`**`()`

Gets the map count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the number of entries in the map

See Also:
`retrieveStatsModule()`, MapStatsModule.getNumEntries(boolean)

---

getInvalidatedSessionsCount

`long` **`getInvalidatedSessionsCount`**`()`

Gets invalidated sessions count

Returns:
the count of invalidated Sessions

---

getActiveSessionsCount

long **getActiveSessionsCount**()

Gets active sessions count

Returns:
the count of active Sessions

---

getMemoryCount

long **getMemoryCount**()

Gets memory count

Returns:
memory count

---

getCacheDiscardsCount

long **getCacheDiscardsCount**()

Gets cache discards count

Returns:
cache discards count

---

getAffinityBreaksCount

long **getAffinityBreaksCount**()

Gets affinity breaks count

Returns:
affinity breaks count

---

getInvalidatedByTimeoutCount

long **getInvalidatedByTimeoutCount**()

Gets invalidated by timeout count

Returns:
count

---

getAccessToNonExistentSessionCount

long **getAccessToNonExistentSessionCount**()

Gets access to non-existent session count

Returns:
count

---

getAccessedSessionsCount

long **getAccessedSessionsCount**()

Gets acessed sessions count

Returns:
count

---

com.ibm.websphere.objectgrid.management

Interface ServerMBean

All Known Subinterfaces:

DynamicServerMBean

`public interface` **`ServerMBean`**

This MBean interface allows a client process to access different attributes about a specific server process.

Since:

WAS XD 6.0.1, XC10

Method Summary

String	**getServerName**() Gets the name of the server associated with this MBean.
void	**modifyServerTraceSpec**(String spec) **Deprecated.** *This is deprecated in version 7.1. See DynamicServerMBean.setTraceSpec(String)*
boolean	**stopServer**() Stops the server associated with this MBean.

Method Detail

getServerName

String **getServerName**()

Gets the name of the server associated with this MBean.

Returns:

the server name

---

stopServer

boolean **stopServer**()

Stops the server associated with this MBean.

Returns:

`true` if server was stopped, `false` if not

---

modifyServerTraceSpec

`void` **`modifyServerTraceSpec`**(`String` spec)

Deprecated. *This is deprecated in version 7.1. See* *`DynamicServerMBean.setTraceSpec(String)`*

Modifies the trace spec for the server associated with this MBean.

Parameters:

`spec` - new trace specification

---

com.ibm.websphere.objectgrid.management

Interface QuorumManagerMBean

public interface **QuorumManagerMBean**

Each catalog service has a QuorumManager and an associated MBean. The QuorumManager monitors and manages the quorum state of the catalog service grid. When quorum is enabled, the QuorumManager for each catalog service process detects when all catalog services in the grid have quorum or not. This MBean allows querying the current quorum state and allows administrators to force quorum when there is a network failure.

```
com.ibm.websphere.objectgrid:type=QuorumManager
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

The following notifications are available:

com.ibm.websphere.objectgrid.quorum.lost

Description:	Catalog service quorum has been lost.
Message:	A translated string identifying the number of active catalog servers and the number in quorum.

com.ibm.websphere.objectgrid.quorum.changed

Description:	The catalog service has quorum, but the number of catalog servers required for quorum has changed.
Message:	A translated string identifying the number of active catalog servers and the number in quorum.

Since:

7.0, XC10

Field Summary

static String	QUORUM_CHANGED_NOTIFICATION
stat	

static [String](#)	**QUORUM_LOST_NOTIFICATION**

Method Summary

[String](#)[]	**getActiveCatalogServerNames**() Retrieves the names of the known active catalog service processes.
int	**getActiveCatalogServers**() Retrieve the known number of active catalog service processes.
int	**getQuorumCatalogServers**() Retrieve the number of catalog service processes required for quorum.
void	**overrideQuorum**() This operation forces surviving catalog service grid processes to reestablish a quorum.

Field Detail

QUORUM_LOST_NOTIFICATION

static final [String](#) **QUORUM_LOST_NOTIFICATION**

See Also:
[Constant Field Values](#)

QUORUM_CHANGED_NOTIFICATION

static final [String](#) **QUORUM_CHANGED_NOTIFICATION**

See Also:
[Constant Field Values](#)

Method Detail

overrideQuorum

void **overrideQuorum**()
throws [Exception](#)

This operation forces surviving catalog service grid processes to reestablish a quorum.

If a portion the catalog service grid fails or is divided due to a network failure, the grid will lose quorum. Once the administrator identifies the failure and the viable portion of the grid, this operation can be invoked on any of the surviving catalog service processes to reestablish a quorum. Reestablishing a quorum will allow the catalog service to

continue to react to failures and topology changes.

Throws:
Exception

---

getActiveCatalogServers

`int` **`getActiveCatalogServers`**`()`

Retrieve the known number of active catalog service processes.

Returns:
the known number of active catalog service processes.

---

getQuorumCatalogServers

`int` **`getQuorumCatalogServers`**`()`

Retrieve the number of catalog service processes required for quorum.

Returns:
the number of catalog service processes required for quorum.

---

getActiveCatalogServerNames

`String[]` **`getActiveCatalogServerNames`**`()`

Retrieves the names of the known active catalog service processes.

Returns:
the names of the known active catalog service processes.
Since:
7.1

---

com.ibm.websphere.objectgrid.management

Interface QueryManagerMBean

public interface **QueryManagerMBean**

This MBean interface allows a client process to perform operations on and get status from an ObjectGrid Query Manager running in a dynamic environment. The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=QueryManager,name=<grid name>,mapset=<mapset name>,partition=<p
artition number>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1.0.5, XC10

Method Summary

double	**getPlanCreationTime**(String query) Gets the query's plan creation time attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryExecutionCount**(String query) Gets the query's execution count attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryExecutionTime**(String query) Gets the query's execution time attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryFailureCount**(String query) Gets the query's failure count attribute loaded up by the `retrieveStatsModule()` method.
double	**getQueryResultCount**(String query) Gets the query's result count attribute loaded up by the `retrieveStatsModule()` method.

com.ibm.websphere.objectgrid.stats.QueryStatsModule	
	retrieveStatsModule(String query) Gets the QueryStatsModule used to retrieve statistics associated with the specified query String

Method Detail

getPlanCreationTime

double **getPlanCreationTime**(String query)

Gets the query's plan creation time attribute loaded up by the retrieveStatsModule() method.

Returns:
the plan creation time for this query in milliseconds

See Also:
retrieveStatsModule(String), QueryStatsModule.getPlanCreationTime(boolean copy)

---

getQueryExecutionTime

double **getQueryExecutionTime**(String query)

Gets the query's execution time attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the execution time for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryExecutionTime(boolean copy)`

---

getQueryExecutionCount

`double` **`getQueryExecutionCount`**`(String query)`

Gets the query's execution count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the execution count for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryExecutionCount(boolean copy)`

---

getQueryResultCount

`double` **`getQueryResultCount`**`(String query)`

Gets the query's result count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the result count for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryResultCount(boolean copy)`

---

getQueryFailureCount

`double` **`getQueryFailureCount`**`(String query)`

Gets the query's failure count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the failure count for this query
See Also:
`retrieveStatsModule(String)`, `QueryStatsModule.getQueryFailureCount(boolean copy)`

---

retrieveStatsModule

`com.ibm.websphere.objectgrid.stats.QueryStatsModule` **`retrieveStatsModule`**`(String query)`

Gets the `QueryStatsModule` used to retrieve statistics associated with the specified query String

Returns:
an QueryStatsModule for statistics associated with the specified query String

---

com.ibm.websphere.objectgrid.management

Interface PlacementServiceMBean

All Superinterfaces:

com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean

---

```
public interface PlacementServiceMBean
extends com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean
```

This MBean interface allows a client process to perform operations on and get status from the PlacementService running in a dynamic environment. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=PlacementService
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static int	**ALL** Constant representing a all shard types
static int	**ASYNCHRONOUS_REPLICA** Constant representing an asynchronous replica shard type.
static int	**PRIMARY** Constant representing a primary shard type.
sta	

static int	**SYNCHRONOUS_REPLICA** Constant representing a synchronous replica shard type.

Fields inherited from interface com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean
HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE, HEARTBEAT_FREQUENCY_LEVEL_RELAXED, HEARTBEAT_FREQUENCY_LEVEL_TYPICAL

Method Summary

String	**balanceShardTypes**(String objectGridName, String mapSetName) The placement service will examine the distribution of primaries and replicas for a given mapSet and attempt (if zone rules and other balancing constraints allow) to achieve a consistent primary to replica ratio across the set of containers.
String	**balanceStatus**(String objectGridName, String mapSetName) Check the balance status (suspended or resumed) for a specified MapSet.
String	**collectContainerStatus**(String objectGridName, String mapSetName) Retrieves the container status for all containers in the domain.
boolean	**enableForPlacement**(String containerName) Re-enables a disabled container for placement.
String	**getCoreGroups**() Gets the coregroup status.
TabularData	**getDisabledForPlacement**() Retrieves TabularData of containers which have been disabled because of the failure of shard placement operations.
boolean	**getInitialized**() Returns whether this instance of the Placement Service finished initialization.

String	**getObjectGridNames**() Gets the names of all ObjectGrids and their mapsets in the domain.
String	**listCoreGroupMembers**(String coreGroupName) List the coregroup members for a given coregroup.
String	**listObjectGridPlacement**(String objectGridName, String mapSetName) List the placement of shards for each container in the domain.
String	**listObjectGridPlacementStatus**(String objectGridName, String mapSetName) List the current placement status.
String	**listPartition**(String objectGridName, String mapSetName, String partitionId) List the partition placement status in the domain.
String	**listShards**(String objectGridName, String mapSetName, String containerName, int mask) List the shard placement status.
String	**listVerifiedRoutingTable**(String objectGridName) **Deprecated.**
String	**replaceLostShards**(String objectGridName, String mapSetName) Lost shards are placed onto the UNREPAIRED container when autoReplaceLostShards is disabled.
String	**resumeBalancing**(String objectGridName, String mapSetName) Execute balancing operation at next opportunity and allow execution of future balancing attempts for the map set specified.
List	**retrieveAllServersJMXAddresses**() Retrieves a List of JMX addresses for all servers that have registered with the placement service.
Str	**retrieveMapSetName**(String gridName, String mapName)

[String](#)	Retrieves the name of the MapSet in which the specified map is defined.
[List](#)	**retrieveServerJMXAddress**([String](#) hostName, [String](#) serverName) Retrieves a List of JMX address strings for a specific host and server name that has registered with the placement service.
[TabularData](#)	**retrieveServerJMXAddressesWithInfo**([String](#) hostName, [String](#) serverName) Retrieves TabularData of JMX address strings for a specific host and server name that has registered with the placement service.
[String](#)	**suspendBalancing**([String](#) objectGridName, [String](#) mapSetName) Prevent future balancing attempts for a specific map set.
[String](#)	**tearDownServers**([String](#)[] servers) Each of the container servers that are passed into this method will be stopped.
[String](#)	**triggerPlacement**([String](#) objectGridName, [String](#) mapSetName) Placement normally occurs implicitly after an event such as an ObjectGrid container starting or stopping.

Methods inherited from interface com.ibm.websphere.objectgrid.management.CoreGroupServiceMBean
getHeartBeatFrequencyLevel, setHeartBeatFrequencyLevel

Field Detail

PRIMARY

static final int **PRIMARY**

Constant representing a primary shard type.

See Also:
[Constant Field Values](#)

SYNCHRONOUS_REPLICA

static final int **SYNCHRONOUS_REPLICA**

Constant representing a synchronous replica shard type.

See Also:
Constant Field Values

ASYNCHRONOUS_REPLICA

```
static final int ASYNCHRONOUS_REPLICA
```

Constant representing an asynchronous replica shard type.

See Also:
Constant Field Values

ALL

```
static final int ALL
```

Constant representing a all shard types

See Also:
Constant Field Values

Method Detail

retrieveServerJMXAddress

```
List retrieveServerJMXAddress(String hostName,
                              String serverName)
```

Retrieves a List of JMX address strings for a specific host and server name that has registered with the placement service.

Parameters:
`hostName` - The name of the host to retrieve the JMX addresses.
`serverName` - The name of the server to retrieve the JMX addresses.
Returns:
the List of all JMX address strings for the specified host and server name.

---

retrieveServerJMXAddressesWithInfo

```
TabularData retrieveServerJMXAddressesWithInfo(String hostName,
                                               String serverName)
```

Retrieves TabularData of JMX address strings for a specific host and server name that has registered with the placement service. Null host or null server names can be used to retrieve several results. Using a null host and null server name will retrieve all of the servers. The TabularData contains CompositeData with the following items, where each CompositeData represents a server:

Item Name	Type	Description
JMXServiceURL	String	JMX Service URL
HostName	String	Host name
ServerName	String	Server Name

Parameters:
`hostName` - The name of the host for which to retrieve JMX addresses. Use null or an empty string to retrieve JMX addresses for any host.
`serverName` - The name of the server for which to retrieve JMX addresses. Use null or an empty string to retrieve JMX addresses for any server.

Returns:
the TabularData of all JMX address strings for the specified host and server name.
Since:
8.5

---

retrieveAllServersJMXAddresses

`List` **`retrieveAllServersJMXAddresses`**`()`

Retrieves a List of JMX addresses for all servers that have registered with the placement service.

Returns:
the List of all servers' JMX addresses

---

collectContainerStatus

`String` **`collectContainerStatus`**`(String objectGridName,`
`String mapSetName)`

Retrieves the container status for all containers in the domain.

The results are returned in the following format:

```
<container name="<container>" zoneName="<zone>" hostName="<host>" serverName="<server>">
  <shard type="<type>" partitionName="<partition>"/>
</container>
```

Parameters:
`objectGridName` - The name of the ObjectGrid for which to get container status.
`mapSetName` - The name of the mapset for which to get the container status.
Returns:
The String status object for all containers in XML form.

---

listObjectGridPlacement

`String` **`listObjectGridPlacement`**`(String objectGridName,`
`String mapSetName)`

List the placement of shards for each container in the domain.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset">
  <container name="<container>" zoneName="<zone>" hostName="<host>" serverName="<server>">
    <shard type="<type>" partitionName="<partition>" reserved="<true>"/>
  /container>
</objectGrid>
```

NOTE: The default value for the "reserved" attribute is false.

Parameters:
`objectGridName` - The name of the ObjectGrid for which to get placement status.
`mapSetName` - The name of the mapset for which to get the placement status.
Returns:
The placement status in XML form.

---

listObjectGridPlacementStatus

```
String listObjectGridPlacementStatus(String objectGridName,
                                      String mapSetName)
```

List the current placement status.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <configuration>
    <attribute name="<placementStrategy>" value="<strategy>"/>
    <attribute name="<numInitialContainers>" value="<num>"/>
    <attribute name="<minSyncReplicas>" value="<min>"/>
    <attribute name="<developmentMode>" value="<mode>"/>
  </configuration>
  <runtime>
    <attribute name="<numContainers>" value="<num>"/>
    <attribute name="<numMachines>" value="<num>"/>
    <attribute name="<numOutstandingWorkItems>" value="<num>"/>
    <attribute name="<numActiveZones>" value="<num>"/>
  </runtime>
</objectGrid>
```

Parameters:

`objectGridName` - The name of the ObjectGrid for which to get placement status.

`mapSetName` - The name of the mapset for which to get the placement status.

Returns:

The placement status in XML form.

---

getCoreGroups

```
String getCoreGroups()
```

Gets the coregroup status.

The results are returned in the following format:

```
<coreGroup name="<coregroup>">
  <coreGroupLeader hostName="<host>" serverName="<server>"/>
  <coreGroupMember hostName="<host>" serverName="<server>"/>
</coreGroup>
```

Returns:

the coregroup status in XML form.

---

listCoreGroupMembers

```
String listCoreGroupMembers(String coreGroupName)
```

List the coregroup members for a given coregroup.

The results are returned in the following format:

```
<coreGroup name="<coregroup>">
  <coreGroupMember hostName="<host>" serverName="<server>"/>
</coreGroup>
```

Parameters:

`coreGroupName` - The name of the coregroup for which to get the members.

Returns:

The coregroup members in XML form.

---

listPartition

```
String listPartition(String objectGridName,
                     String mapSetName,
                     String partitionId)
```

List the partition placement status in the domain. The results are returned in the following format:

```
<partition name="<partition>">
  <shard type="<type>" containerName="<container>" hostName="<host>" serverName="<server>"/>
</partition>
```

Parameters:

`objectGridName` - The name of the ObjectGrid for which to get placement status.
`mapSetName` - The name of the mapset for which to get the placement status.
`partitionId` - The name of the partition for which to get the placement status.

Returns:

The partition placement status in the XML form.

---

listShards

```
String listShards(String objectGridName,
                  String mapSetName,
                  String containerName,
                  int mask)
```

List the shard placement status.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <container name="<container>" zoneName="<zone>" hostName="<host>" serverName="<server>">
    <shard type="<type>" partitionName="<partition>"/>
  </container>
</objectGrid>
```

Parameters:

`objectGridName` - The name of the ObjectGrid for which to get placement status.
`mapSetName` - The name of the mapset for which to get the placement status.
`containerName` - The name of the container for which to get the placement status. If empty string (""), get shard placement for all containers.
`mask` - The Integer mask to determine for which shard types to get status.

Returns:

The shard placement status in XML form.

See Also:

ALL, PRIMARY, SYNCHRONOUS_REPLICA, ASYNCHRONOUS_REPLICA

---

getObjectGridNames

```
String getObjectGridNames()
```

Gets the names of all ObjectGrids and their mapsets in the domain.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>"/>
```

Returns:

the names of all ObjectGrids and their mapsets in the domain in XML form.

---

replaceLostShards

```
String replaceLostShards(String objectGridName,
                         String mapSetName)
```

Lost shards are placed onto the UNREPAIRED container when autoReplaceLostShards is disabled. Shards on the UNREPAIRED will not be placed until this method is called.

Calling this method will move shards off the UNREPAIRED container onto the UNASSIGNED container.

Balance and placement operations will be queued up for the MapSet specified. These operations will execute when all outstanding placement work from previous events has completed.

The string returned is an XML representation of the shards that moved as a result of the call to this method.

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <shard type="<type>" partitionName="<partition>">
     <currentContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<se
rver>"/>
     <previousContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<s
erver>"/>
   </shard>
 </objectGrid>
```

The returned XML will look as follows when no shards have been moved:

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <!-- No shards were moved -->
 </objectGrid>
```

Parameters:

`objectGridName` - replace lost shards for this ObjectGrid

`mapSetName` - replace lost shards for this MapSet

Returns:

An XML String containing shards that have moved

Since:

WAS XD 6.1.0.5

---

triggerPlacement

```
String triggerPlacement(String objectGridName,
                        String mapSetName)
```

Placement normally occurs implicitly after an event such as an ObjectGrid container starting or stopping.

Calling this method will trigger a placement operation for the ObjectGrid and MapSet specified.

Under normal circumstances, the `numInitialContainers` attribute (in the deployment policy) must be met in order for placement to occur. However, when this method is called, the numInitialContainers value is ignored.

The string returned is an XML representation of the shards that moved as a result of the call to this method.

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <shard type="<type>" partitionName="<partition>">
     <currentContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<se
rver>"/>
     <previousContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<s
erver>"/>
   </shard>
 </objectGrid>
```

The returned XML will look as follows when no shards have been moved:

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <!-- No shards were moved -->
 </objectGrid>
```

Parameters:

`objectGridName` - trigger placement for this ObjectGrid
`mapSetName` - trigger placement for this MapSet

Returns:

An XML String containing shards that have moved

Since:

WAS XD 6.1.0.5

See Also:

`ObjectGridDeployment.addMapSet(com.ibm.websphere.objectgrid.deployment.MapSet)`, `MapSet.setNumInitialContainers(int)`

---

tearDownServers

`String` **tearDownServers**(`String`[] servers)

Each of the container servers that are passed into this method will be stopped. If the server cannot be reached, all of the server's artifacts will be removed.

Use this method if servers are found to be in a corrupt state or bindings need to be cleared from the catalog server.

The string returned is an XML representation of the results of the attempt to tear down each of the servers. If the command is successful, the XML will look as follows:

```
<domain name="<domain>">
  <server name="<server>" tearDownSuccessful="true"/>
  <server name="<server>" tearDownSuccessful="true"/>
</domain>
```

If the command is not successful, the string will look as follows (where the exception element is only present if an exception is part of the failure):

```
<domain name="<domain>">
  <server name="<server>" tearDownSuccessful="false" reason="<String>">
     <exception type="<String>" message="<String>" stack="<String>"/>
  </server>
</domain>
```

Parameters:

`servers` - String array of servers to tear down.

Returns:

An XML String containing the results of tear down attempts.

Since:

WAS XD 6.1.0.5 FIX2

---

listVerifiedRoutingTable

```
@Deprecated
String listVerifiedRoutingTable(String objectGridName)
```

Deprecated.

This method is deprecated. The com.ibm.websphere.objectgrid.client.RouteTableValidation utility replaces this method.

Calling this method will return an XML string of the current known routing table. The Placement service will contact each shard and return state on whether it was able to verify that's shard's existence. All shards will be included in the XML doc, whether they were reachable or not. The user can use the `reachable` attribute below to filter valid or invalid shards.

```
 <objectGrid name="<objectgrid>" name="<name>">
   <primary zone="<zone>"> partition="<partition>"> state="<reachable>"> ipaddress="<ipaddres
s>">
     <replica zone="<zone>"> partition="<partition>"> state="<reachable>"> ipaddress="<ipaddr
ess>">
   </primary>
 </objectGrid>
```

Parameters:

`objectGridName` - retrieve routing table for this ObjectGrid

Returns:

An XML String containing a pre-verified routing table

Since:

WAS XD 6.1.0.5 FIX2

---

retrieveMapSetName

```
String retrieveMapSetName(String gridName,
                          String mapName)
```

Retrieves the name of the MapSet in which the specified map is defined.

Parameters:

`gridName` - the name of the ObjectGrid

`mapName` - the name of the map

Returns:

the name of the MapSet in which the specified map is defined.

Since:

7.0

---

getInitialized

```
boolean getInitialized()
```

Returns whether this instance of the Placement Service finished initialization.

Returns:

true if initialized, false if not initialized.

Since:

8.5

---

balanceShardTypes

```
String balanceShardTypes(String objectGridName,
                         String mapSetName)
```

The placement service will examine the distribution of primaries and replicas for a given mapSet and attempt (if zone rules and other balancing constraints allow) to achieve a consistent primary to replica ratio across the set of containers.

If the number of primaries or the number of replicas do not divide evenly across the containers, some tolerance must be allowed for the ratio to differ from container to container. However, the difference in the number of primaries from one container to the next will not be greater than 1. Similarly, the difference in the number of replicas from one container to the next will not be greater than 1.

Null arguments are not allowed as input to this method.

The results are returned in the following format:

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <shard type="<type>" partitionName="<partition>">
     <currentContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<se
rver>"/>
     <previousContainer name="<container>" zoneName="<zone>" hostName="<host>" serverName="<s
erver>"/>
   </shard>
 </objectGrid>
```

If no shards were moved or a problem was encountered attempting to execute this method, no shard elements will appear in the XML output. A detail element will appear instead. The message attribute will have further information.

```
 <objectGrid name="<objectgrid>" mapSetName="<mapset>">
   <detail message="<message>" />
 </objectGrid>
```

Parameters:

`objectGridName` - the grid

`mapSetName` - the map set within the grid

Returns:

An XML String containing the results of the attempt to redistribute shards for better primary/replica balance

Since:

7.1.1

---

suspendBalancing

```
String suspendBalancing(String objectGridName,
                        String mapSetName)
```

Prevent future balancing attempts for a specific map set. Balancing work that is in progress will be allowed to complete.

Other placement activities are allowed to execute while balancing is suspended.

- shard promotion due to container loss
- shard role swap
- shard reservation
- triggerPlacement
- replaceLostShards

Balancing will remain suspended until it is resumed by calling `resumeBalancing(String, String)`.

Null arguments are not allowed as input to this method.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>"/>
</objectGrid>
```

Additionally, an optional detail element may be contained within the suspendBalancing element. The detail element will include additional data regarding execution of this method. The XML result will be in the following format when a detail element is included:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>">
    <detail message="<message>" />
  <suspendBalancing/>
</objectGrid>
```

Parameters:

`objectGridName` - suspend balancing for the map set specified within this ObjectGrid
`mapSetName` - suspend balancing for this map set

Returns:

An XML String containing the results of the attempt to suspend balancing

Since:

7.1.0.3

See Also:

`resumeBalancing(String, String)`

---

resumeBalancing

```
String resumeBalancing(String objectGridName,
                       String mapSetName)
```

Execute balancing operation at next opportunity and allow execution of future balancing attempts for the map set specified. Balancing is executed in reaction to key placement events. Such events include containers starting and containers stopping.

By default, balancing work is executed unless `suspendBalancing(String, String)` has been called for the map set.

Null arguments are not allowed as input to this method.

The results are returned in the following format:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>"/>
</objectGrid>
```

Additionally, an optional detail element may be contained within the suspendBalancing element. The detail element will include additional data regarding execution of this method. The XML result will be in the following format when a detail element is included:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <suspendBalancing currentValue="<currentValue>" previousValue="<previousValue>">
    <detail message="<message>" />
  <suspendBalancing/>
</objectGrid>
```

Parameters:

`objectGridName` - resume balancing for the map set specified within this ObjectGrid

`mapSetName` - resume balancing for this map set

Returns:

An XML String containing the results of the attempt to resume balancing

Since:

7.1.0.3

See Also:

suspendBalancing(String, String)

---

balanceStatus

```
String balanceStatus(String objectGridName,
                     String mapSetName)
```

Check the balance status (suspended or resumed) for a specified MapSet.

Null arguments are not allowed as input to this method.

The string returned is an XML representation of the balance status. The XML will look as follows:

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <balanceStatus suspended="<suspended>" />
</objectGrid>
```

Additionally, an optional detail element may be contained within the balanceStatus element. When balancing has been pre-suspended, the message attribute of the detail element will contain the following message.

```
<objectGrid name="<objectgrid>" mapSetName="<mapset>">
  <balanceStatus suspended="true" >
    <detail message="Balancing has been pre-suspended for this mapSet." />
  </balanceStatus>
</objectGrid>
```

Parameters:

`objectGridName` - check balance status for the map set specified within this ObjectGrid

`mapSetName` - check balance status for this map set

Returns:

An XML String containing the balance status

Since:

7.1.1

See Also:

suspendBalancing(String, String), resumeBalancing(String, String)

---

enableForPlacement

```
boolean enableForPlacement(String containerName)
```

Re-enables a disabled container for placement. A container may become disabled because of a falure to place a shard into into the container.

Use the getDisabledForPlacement() attribute to determine which containers are disabled.

Parameters:

`containerName` - The name of the container to re-enable.

Returns:

Answers true if the container's status was changed from disabled to enabled, false if

the container was already enabled for placement.

Since:

8.6, XC10 2.5

---

getDisabledForPlacement

[TabularData](#) **getDisabledForPlacement**()

Retrieves TabularData of containers which have been disabled because of the failure of shard placement operations. The TabularData contains CompositeData with the following items, where each CompositeData represents a container:

Item Name	Type	Description
Container	String	A container that has been disabled for placement

Returns:

A TabularData of the names of disabled containers.

Since:

8.6, XC10 2.5

---

com.ibm.websphere.objectgrid.management

Interface PlacementMediationServiceMBean

public interface **PlacementMediationServiceMBean**

This MBean interface allows a client process to perform operations on and get status from the PlacementMediationService running in a dynamic environment. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=PlacementMediationService
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
7.1, XC10

Method Summary

CompositeData	**dismissLink**(String foreignDomain) Dismiss a previously established link with the foreign domain specified.
CompositeData	**establishLink**(String foreignDomain, String endPoints) Establish a link between this domain and the foreign domain specified.
TabularData	**getLinkedDomains**() Retrieve the foreign domains that have an active link with the local domain.

[a](#)	
[Tabular Data](#)	**[getLinkedDomainsWithGrids](#)**() Retrieve the foreign domains that have an active link with the local domain and the map sets eligible for linking.

Method Detail

establishLink

[CompositeData](#) **establishLink**([String](#) foreignDomain,
[String](#) endPoints)

Establish a link between this domain and the foreign domain specified. This is functionally equivalent to providing the foreign domain and its end points in the server properties file at server startup time.

Domains that are linked will share placement with each other. When compatible map sets are detected within linked domains, a multi-primary topology will be achieved. Data written to a primary is either domain will be asynchronously replicated to the other domain.

The result is a CompositeData that includes the following items:

Item Name	Type	Description
Result	String	The result of the attempt.
StatusBefore	String	The status of the link before the attempt was made to establish the link.
StatusAfter	String	The status of the link after the attempt was made to establish the link.

Parameters:

`foreignDomain` - the name of the foreign domain
`endPoints` - end points of the foreign domain

Returns:

CompositeData representing the status of the attempt to link with the foreign domain

See Also:

`CatalogServerProperties.setForeignDomains(String)`,
`CatalogServerProperties.setDomainEndPoints(String, String)`

---

dismissLink

[CompositeData](#) **dismissLink**([String](#) foreignDomain)

Dismiss a previously established link with the foreign domain specified. Any map sets that were participating in a multi-primary topology will be disconnected from each other. Data will no longer be replicated from between domains.

The result is a CompositeData that includes the following items:

Item Name	Type	Description
Result	String	The result of the attempt. Can be one of: SUCCESS, FAILURE, NOP
Status Before	String	The status of the link before the attempt was made to dismiss the link. Can be one of: LINKED, ESTABLISHING_LINK, UNLINKED, DISMISSING_LINK
Status After	String	The status of the link after the attempt was made to dismiss the link. Can be one of: LINKED, ESTABLISHING_LINK, UNLINKED, DISMISSING_LINK

Parameters:

`foreignDomain` - the name of the foreign domain

Returns:

CompositeData representing the status of the attempt to dismiss the link with the foreign domain

---

getLinkedDomains

TabularData **getLinkedDomains**()

Retrieve the foreign domains that have an active link with the local domain.

The result is a TabularData where each row is a CompositeData that includes the following items:

Item Name	Type	Description
Domain	String	The name of the foreign domain linked to the local domain.

Returns:

TabularData representing the foreign domains linked to this domain

---

getLinkedDomainsWithGrids

TabularData **getLinkedDomainsWithGrids**()

Retrieve the foreign domains that have an active link with the local domain and the map sets eligible for linking.

The result is a TabularData where each row is a CompositeData that includes the following items:

Item Name	Type	Description
Domain	String	The name of the foreign domain linked to the local domain.
ObjectGrid	String	The name of the ObjectGrid that is compatible with the foreign domain.
MapSet	String	The name of the map set that is compatible with the foreign domain.

Returns:

TabularData representing the foreign domains and eligible map sets linked to this domain

Since:

8.6, XC10 2.5

com.ibm.websphere.objectgrid.management

Interface ObjectGridMBean

public interface **ObjectGridMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific ObjectGrid on a server process. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGrid,name=<objectgrid>,mapset=<mapset>,partition=<partition id>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
WAS XD 6.0.1, XC10

Method Summary

String	**getContainerName**() Gets the name of the container name containing the replication group member for the ObjectGrid associated with this MBean.
long	**getCurrentRevision**() Retrieves the current revision number of this ObjectGrid shard.
String	**getDomainName**() Retrieves the domain name of this ObjectGrid shard.
TabularData	**getKnownRevisions**() An ObjectGrid shard may exist over several different lifetimes.
String	**getLifetimeId**() Retrieves the lifetime id for this ObjectGrid shard.

[ng](#)	
[String](#)	**getObjectGridName**() Gets the name of the ObjectGrid associated with this MBean.
long	**getOGCount**() Gets the ObjectGrid count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getOGMaxTranTime**() Gets the maximum transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getOGMeanTranTime**() Gets the mean transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getOGMinTranTime**() Gets the minimum transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
[String](#)	**getOGStatsModule**() Gets a string representation of the `OGStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`
long	**getOGTotalTranTime**() Gets the total transaction time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getOGTransPerSecond**() Transactions per second attribute loaded up by the retrieveStatsModule call.
[TabularData](#)	**getPrimaryShardLinks**() Get the shard's list of foreign or domestic linked primaries.
[String](#)	**getServerName**() Gets the name of the server containing the replication group member for the ObjectGrid associated with this MBean.
com.	

com.ibm.websphere.objectgrid.stats.OGStatsModule
retrieveStatsModule() Gets the OGStatsModule used to retrieve statistics associated with the ObjectGrid for this MBean.

Method Detail

retrieveStatsModule

com.ibm.websphere.objectgrid.stats.OGStatsModule **retrieveStatsModule**()

Gets the OGStatsModule used to retrieve statistics associated with the ObjectGrid for this MBean.

Returns:
an OGStatsModule for statistics associated with this ObjectGrid

See Also:
OGStatsModule

---

getObjectGridName

String **getObjectGridName**()

Gets the name of the ObjectGrid associated with this MBean.

Returns:
name of the ObjectGrid

---

getServerName

String **getServerName**()

Gets the name of the server containing the replication group member for the ObjectGrid associated with this MBean.

Returns:
the name of server containing the replication group member for the ObjectGrid associated with this MBean.

---

getContainerName

String **getContainerName**()

Gets the name of the container name containing the replication group member for the ObjectGrid associated with this MBean.

Returns:
the name of container containing the replication group member for the ObjectGrid associated with this MBean.
Since:
8.5

---

getOGStatsModule

String **getOGStatsModule**()

Gets a string representation of the `OGStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`

Returns:
String form of OGStatsModule
See Also:
retrieveStatsModule(), OGStatsModule

---

getOGCount

long **getOGCount**()

Gets the ObjectGrid count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the number of transactions
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGMaxTranTime

long **getOGMaxTranTime**()

Gets the maximum transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the maximum transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGMinTranTime

long **getOGMinTranTime**()

Gets the minimum transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the minimum transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGMeanTranTime

double **getOGMeanTranTime**()

Gets the mean transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the mean transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGTotalTranTime

long **getOGTotalTranTime**()

Gets the total transaction time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
the total transaction time for the ObjectGrid in milliseconds
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getOGTransPerSecond

long **getOGTransPerSecond**()

Transactions per second attribute loaded up by the retrieveStatsModule call. retrieveStatsModule() or refreshStatsModule() method.

Returns:
the transactions per second for the ObjectGrid
See Also:
retrieveStatsModule(), OGStatsModule.getTransactionTime(String, boolean)

---

getCurrentRevision

long **getCurrentRevision**()

Retrieves the current revision number of this ObjectGrid shard.

Returns:
the current revision number of this ObjectGrid shard.
Since:
7.1

---

getDomainName

String **getDomainName**()

Retrieves the domain name of this ObjectGrid shard.

Returns:
the name of the domain name of this ObjectGrid shard.
Since:
7.1

---

getLifetimeId

String **getLifetimeId**()

Retrieves the lifetime id for this ObjectGrid shard.

Returns:
the lifetime id for this ObjectGrid shard.
Since:
7.1

---

getKnownRevisions

TabularData **getKnownRevisions**()

An ObjectGrid shard may exist over several different lifetimes. As such, each shard instance will have a unique lifetime id and revision number associated with it. This method returns a TabularData object representing the known history of revision numbers for each lifetime. Each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Domain	String	The domain name of this ObjectGrid shard.	7.1
Server	String	The name of the server owning the lifetime id.	8.5
LifetimeId	String	The lifetime id of this ObjectGrid shard.	7.1
Revision	Long	The revision of this ObjectGrid shard.	7.1

Returns:
TabularData representing the known lifetimes and revisions of this shard.
Throws:
OpenDataException
Since:
7.1
See Also:
TabularData

---

getPrimaryShardLinks

TabularData **getPrimaryShardLinks**()

Get the shard's list of foreign or domestic linked primaries.

This method returns a TabularData object representing the current state of each primary shard link.

Each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description
RemoteDomain	String	The catalog service domain name of the remote primary shard..
RemoteContainer	String	The container name of the remote primary shard.
Status	String	The status of the link. Valid states include: online and recovery.

Returns:
TabularData representing the linked primaries
Since:
7.1.1

---

com.ibm.websphere.objectgrid.management

Interface MapMBean

```
public interface MapMBean
```

This MBean interface allows a client process to access different attributes and statistical data about a specific map on a server process. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectMap,name=<map>,partition=<partition id>,objectgrid=<objectgrid>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.0.1, XC10

Method Summary

String	**getContainerName**() Gets the name of the container containing the replication group member for the map associated with this MBean.
double	**getMapBatchUpdateMaxTime**() Gets the maximum batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapBatchUpdateMeanTime**() Gets the mean batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapBatchUpdateMinTime**() Gets the minimum batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapBatchUpdateTotalTime**() Gets the total batch update time attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.

long	**getMapCountStatistic**() Gets the map count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getMapGetCountStatistic**() Gets the get count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
long	**getMapHitCountStatistic**() Gets the hit count attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
double	**getMapHitRateStatistic**() Gets the hit rate attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
String	**getMapName**() Gets the name of the map associated with this MBean.
String	**getMapStatsModule**() Gets a string representation of the `MapStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`
long	**getMapUsedBytes**() Gets the used bytes attribute loaded up by the `retrieveStatsModule()` or `refreshStatsModule()` method.
String	**getObjectGridName**() Gets the name of the ObjectGrid containing the map associated with this MBean.
int	**getPartitionId**() Retrieves the partition identifier for this map instance.
String	**getServerName**() Gets the name of the server containing the replication group member for the map associated with this MBean.
TabularData	**retrieveEntries**(String regex) Operation to iterate through all of the entries in this map, convert the key to string form, then match the string against the regular expression if passed, finally return the matching entries.

com.ibm.websphere.objectgrid.stats.MapStatsModule

retrieveStatsModule()
Gets the MapStatsModule used to retrieve statistics associated with the map for this MBean.

Method Detail

retrieveStatsModule

com.ibm.websphere.objectgrid.stats.MapStatsModule **retrieveStatsModule**()

Gets the MapStatsModule used to retrieve statistics associated with the map for this MBean.

Returns:
A MapStatsModule for statistics associated with this map.

See Also:
MapStatsModule

---

getMapName

String **getMapName**()

Gets the name of the map associated with this MBean.

Returns:
The name of the map.

---

getObjectGridName

String **getObjectGridName**()

Gets the name of the ObjectGrid containing the map associated with this MBean.

Returns:
The name of the ObjectGrid for the map associated with this MBean.

---

getServerName

String **getServerName**()

Gets the name of the server containing the replication group member for the map associated with this MBean.

Returns:
The name of server containing the replication group member for the map associated with this MBean.

---

getMapStatsModule

String **getMapStatsModule**()

Gets a string representation of the `MapStatsModule attributes loaded up by the retrieveStatsModule() or refreshStatsModule() method.`

Returns:
The String form of MapStatsModule
See Also:
retrieveStatsModule(), MapStatsModule

---

getMapCountStatistic

long **getMapCountStatistic**()

Gets the map count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The number of entries in the map.
See Also:
retrieveStatsModule(), MapStatsModule.getNumEntries(boolean)

---

getMapHitRateStatistic

double **getMapHitRateStatistic**()

Gets the hit rate attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The hit rate for the map.
See Also:
retrieveStatsModule(), MapStatsModule.getHitRate(boolean)

---

getMapGetCountStatistic

long **getMapGetCountStatistic**()

Gets the get count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The get count for the map.
Since:
7.1
See Also:
retrieveStatsModule(), MapStatsModule.getHitRate(boolean)

---

getMapUsedBytes

long **getMapUsedBytes**()

Gets the used bytes attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

The used bytes statistics are accurate only when you are using simple objects or the COPY_TO_BYTES copy mode.

Returns:
The number of bytes in use by the map.
Since:
7.1
See Also:
retrieveStatsModule(), MapStatsModule.getUsedBytes(boolean)

---

getMapHitCountStatistic

long **getMapHitCountStatistic**()

Gets the hit count attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The hit count for the map.
Since:
7.1
See Also:
retrieveStatsModule(), MapStatsModule.getHitRate(boolean)

---

getMapBatchUpdateMeanTime

double **getMapBatchUpdateMeanTime**()

Gets the mean batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The mean batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getMapBatchUpdateMaxTime

double **getMapBatchUpdateMaxTime**()

Gets the maximum batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The maximum batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getMapBatchUpdateMinTime

double **getMapBatchUpdateMinTime**()

Gets the minimum batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The minimum batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getMapBatchUpdateTotalTime

double **getMapBatchUpdateTotalTime**()

Gets the total batch update time attribute loaded up by the retrieveStatsModule() or refreshStatsModule() method.

Returns:
The total batch update time for the map in milliseconds.
See Also:
retrieveStatsModule(), MapStatsModule.getBatchUpdateTime(boolean)

---

getPartitionId

int **getPartitionId**()

Retrieves the partition identifier for this map instance.

Returns:
The partition identifier.
Since:
WAS XD 6.1.0.4

---

getContainerName

String **getContainerName**()

Gets the name of the container containing the replication group member for the map associated with this MBean.

Returns:
The name of container containing the replication group member for the map associated with this MBean.
Since:
8.5

---

retrieveEntries

TabularData **retrieveEntries**(String regex)

Operation to iterate through all of the entries in this map, convert the key to string form, then match the string against the regular expression if passed, finally return the matching entries. This method could potentially return a very large data structure so care should be taken to ensure the regular expression will reduce the number of keys appropriately.

Each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description
KeyName	String	The domain name of this ObjectGrid shard.
LifetimeIndex	Short	The lifetime index for revisioning.
Revision	Long	The revision number of the last update.

Parameters:
regex - the regular expression to apply to the String form of the key. It should be used in narrowing the entries returned. If null, all entries are returned.

Returns:

A table of entries containing the user readable (String) form of the key and some meta information about the entry.

Since:

7.1.1

---

com.ibm.websphere.objectgrid.management

Interface HashIndexMBean

public interface **HashIndexMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific HashIndex on a server process. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=HashIndex,name=<index-name>,partition=<partition id>,objectgrid
=<objectgrid>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1.0.5, XC10

Method Summary

double	**getBatchUpdateCount**() Gets the index's batchupdate count attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindCollisionCount**() Gets the index's collision count attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindCount**() Gets the index's find count attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindDurationTime**() Gets the index's find duration time attribute loaded up by the `retrieveStatsModule()` method.
double	**getFindFailureCount**() Gets the index's failure count attribute loaded up by the `retrieveStatsModule()` method.

double	**getFindResultCount**() Gets the index's result count attribute loaded up by the `retrieveStatsModule()` method.
String	**getParentMapName**() Gets the index's parent map name
com.ibm.websphere.objectgrid.stats.HashIndexStatsModule	**retrieveStatsModule**() Gets the `HashIndexStatsModule` used to retrieve statistics associated with this index

Method Detail

getParentMapName

`String` **getParentMapName**()

Gets the index's parent map name

Returns:
the name of the map which this index belongs to

---

retrieveStatsModule

`com.ibm.websphere.objectgrid.stats.HashIndexStatsModule` **retrieveStatsModule**()

Gets the `HashIndexStatsModule` used to retrieve statistics associated with this index

Returns:
an HashIndexStatsModule for statistics associated with this index
See Also:
`HashIndexStatsModule`

---

getFindCount

`double` **getFindCount**()

Gets the index's find count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the find operation's invocation count for this index
See Also:
`retrieveStatsModule()`, `HashIndexStatsModule.getFindCount(boolean copy)`

---

getFindDurationTime

`double` **getFindDurationTime**()

Gets the index's find duration time attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the find call's duration time for this index in milliseconds
See Also:
`retrieveStatsModule()`, `HashIndexStatsModule.getFindDurationTime(boolean copy)`

---

getFindResultCount

`double` **getFindResultCount**()

Gets the index's result count attribute loaded up by the `retrieveStatsModule()` method.

Returns:
the result count for this index and find operation
See Also:
`retrieveStatsModule()`, `HashIndexStatsModule.getFindResultCount(boolean copy)`

---

getFindFailureCount

`double` **getFindFailureCount**()

Gets the index's failure count attribute loaded up by the `retrieveStatsModule()` method.

Returns:

the failure count for this index and find operation

See Also:

retrieveStatsModule(), HashIndexStatsModule.getFindFailureCount(boolean copy)

---

getFindCollisionCount

double **getFindCollisionCount**()

Gets the index's collision count attribute loaded up by the retrieveStatsModule() method.

Returns:

the collision count for this index and find operation

See Also:

retrieveStatsModule(), HashIndexStatsModule.getFindCollisionCount(boolean copy)

---

getBatchUpdateCount

double **getBatchUpdateCount**()

Gets the index's batchupdate count attribute loaded up by the retrieveStatsModule() method.

Returns:

the doBatchUpdate method's invocation count for this index

See Also:

retrieveStatsModule(), MapIndexPlugin.doBatchUpdate(TxID txid, LogSequence sequence), HashIndexStatsModule.getBatchUpdateCount(boolean copy)

---

com.ibm.websphere.objectgrid.management

Interface DynamicServerMBean

All Superinterfaces:

ServerMBean

```
public interface DynamicServerMBean
extends ServerMBean
```

This MBean interface allows a client process to access different attributes about a specific server process in a dynamic environment. The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGridServer,name=<server>,host=<host>,ogServerName=<server
>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

The following notifications are available:

`og.server.container.coregroup.membership.change`

Description:	All core group membership changes detected by the server's core group manager.
UserData:	The number of members in the core group.
Message:	The name of the core group.
Since:	6.1 FIX 3

`com.ibm.websphere.objectgrid.log`

Description:	All log messages detected by the log notification filter. See the `setLogNotificationFilter(String)` attribute.
UserData:	A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The log record severity level.	8.6
LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The originating log record class name.	8.6
SourceMethodName	String	The originating log record method name.	8.6
SequenceNumber	Long	The log record sequence number	8.6
ThreadID	Integer	The originating log record thread id.	8.6
ThrownMessage	Strin	The thrown exception message or empty string, if	8.6

g not available.

Message: The log message.

Since: 8.6

`com.ibm.websphere.objectgrid.ffdc`

Description: All first-failure data captured by the grid server.

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
SourceID	String	The source code generating the exception.	8.6
ProbeID	String	The source code location associated with the exception captured.	8.6
ExceptionName	String	The exception captured.	8.6
Count	Integer	The current occurrence count for the specified exception.	8.6
DateOfFirstOccurrence	String	The exception's first occurrence time stamp.	8.6
Label	String	The label associated with the captured exception occurrence.	8.6

Message: A notification was generated on the server for a new exception.

Since: 8.6

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static String	**COUNT**
static String	**DATE_OF_FIRST_OCCURRENCE**
static String	**EXCEPTION_NAME**

[String](#)	
static [String](#)	**FFDC_INCIDENT_DATA**
static [String](#)	**FFDC_NOTIFICATION**
static [String](#)	**LABEL**
static [String](#)	**LEVEL_NAME**
static [String](#)	**LOG_MESSAGE_NOTIFICATION** The constant identifying the notification type for log messages.
static [String](#)	**LOG_RECORD_DATA**

static String	**LOGGER_NAME**
static String	**PROBE_ID**
static String	**SEQUENCE_NUMBER**
static String	**SERVER_COREGROUP_MEMBERSHIP_CHANGE** A constant identifying the notification type for all core group membership changes detected by this server.
static String	**SOURCE_CLASS_NAME**
static String	**SOURCE_ID**

String	
static String	**SOURCE_METHOD_NAME**
static String	**THREAD_ID**
static String	**THROWN_MESSAGE**

Method Summary

void	**checkFFDCNotification**() Triggers a simulated exception to be captured as a first-failure data capture (FFDC) event (and subsequently broadcasted as a JMX notification), as a means to test and verify the monitoring being enabled on the server.
void	**checkLoggingNotification**() Generates a set of log records with various severity levels, providing a simple means to test and verify when the JMX notification monitoring being enabled on the grid server.
int	**getAvailableProcessors**() Returns the number of available processors for the JVM hosting this server.
CompositeData	**getEnvironmentInfo**() Retrieve the environment information for the server (host name, WebSphere eXtreme Scale version, and additional information).
lo	**getFreeMemory**() Returns the available memory in bytes for the JVM hosting this server.

ng	
String	**getHostName**() Returns the host name for this process.
String	**getLogNotificationFilter**() Retrieves current regular expression filter being applied while screening log record messages before being broadcasted as JMX notifications by the grid server as a JMX Notification of type:
long	**getMaxMemory**() Returns the maximum memory in bytes for the JVM hosting this server.
TabularData	**getPrimaryRevisions**() Provides revisions for all primary shards in the container in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:
TabularData	**getRevisions**() Provides revisions for all shards in the server in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:
boolean	**getSafeToShutdown**() Returns true if a replica exists for each primary hosted on this server.
String	**getStatsSpec**() Retrieve the current statistics specification for the server.
long	**getTotalMemory**() Returns the total memory in bytes for the JVM hosting this server.
String	**getTraceSpec**() Retrieve the current trace specification for the server.

String	**getZoneName**() Returns the zone name for this process
void	**setLogNotificationFilter**(String regExFilter) Sets the regular expression filter to be applied in screening log record messages before being broadcasted by the grid server as a JMX Notification of type:
void	**setStatsSpec**(String statsSpec) Set the statistics specification for the server.
void	**setTraceSpec**(String traceSpec) Set the trace specification for the server.

Methods inherited from interface com.ibm.websphere.objectgrid.management.ServerMBean
getServerName, modifyServerTraceSpec, stopServer

Field Detail

SERVER_COREGROUP_MEMBERSHIP_CHANGE

static final String **SERVER_COREGROUP_MEMBERSHIP_CHANGE**

A constant identifying the notification type for all core group membership changes detected by this server.

Since:
6.1 FIX 3

See Also:
Constant Field Values

LOG_MESSAGE_NOTIFICATION

static final String **LOG_MESSAGE_NOTIFICATION**

The constant identifying the notification type for log messages.

Since:
8.6, XC10 2.5

See Also:
Constant Field Values

FFDC_NOTIFICATION

static final String **FFDC_NOTIFICATION**

See Also:
Constant Field Values

LOG_RECORD_DATA

static final String **LOG_RECORD_DATA**

See Also:
Constant Field Values

THROWN_MESSAGE

static final String **THROWN_MESSAGE**

See Also:
Constant Field Values

THREAD_ID

static final String **THREAD_ID**

See Also:
Constant Field Values

SEQUENCE_NUMBER

static final String **SEQUENCE_NUMBER**

See Also:
Constant Field Values

SOURCE_METHOD_NAME

static final String **SOURCE_METHOD_NAME**

See Also:
Constant Field Values

SOURCE_CLASS_NAME

static final String **SOURCE_CLASS_NAME**

See Also:
Constant Field Values

LOGGER_NAME

static final String **LOGGER_NAME**

See Also:
Constant Field Values

LEVEL_NAME

static final String **LEVEL_NAME**

See Also:
Constant Field Values

FFDC_INCIDENT_DATA

static final String **FFDC_INCIDENT_DATA**

See Also:
Constant Field Values

SOURCE_ID

static final String **SOURCE_ID**

See Also:
Constant Field Values

PROBE_ID

static final String **PROBE_ID**

See Also:
Constant Field Values

LABEL

static final String **LABEL**

See Also:
Constant Field Values

DATE_OF_FIRST_OCCURRENCE

static final String **DATE_OF_FIRST_OCCURRENCE**

See Also:
Constant Field Values

COUNT

static final String **COUNT**

See Also:
Constant Field Values

EXCEPTION_NAME

static final String **EXCEPTION_NAME**

See Also:
Constant Field Values

Method Detail

getAvailableProcessors

int **getAvailableProcessors**()

Returns the number of available processors for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.availableProcessors()]

---

getFreeMemory

`long` **`getFreeMemory`**`()`

Returns the available memory in bytes for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.freeMemory()]

---

getMaxMemory

`long` **`getMaxMemory`**`()`

Returns the maximum memory in bytes for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.maxMemory()]

---

getTotalMemory

`long` **`getTotalMemory`**`()`

Returns the total memory in bytes for the JVM hosting this server.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[Runtime.totalMemory()]

---

getHostName

[String] **`getHostName`**`()`

Returns the host name for this process.

Returns:
The answer from the Runtime call on the JVM hosting this server.
See Also:
[InetAddress.getHostName()]

---

getZoneName

[String] **`getZoneName`**`()`

Returns the zone name for this process

Returns:

the zone name that was included in the properties used to start the server or DefaultZone if no zone name was used

---

getSafeToShutdown

`boolean` **`getSafeToShutdown`**`()`

Returns true if a replica exists for each primary hosted on this server. Returns false if the server has the only copy of data.

Returns:
If server is safe to shutdown.

---

getStatsSpec

`String` **`getStatsSpec`**`()`

Retrieve the current statistics specification for the server.

Returns:
a string representation of the statistics specification.
Since:
7.1
See Also:
`StatsSpec`

---

setStatsSpec

`void` **`setStatsSpec`**`(String statsSpec)`

Set the statistics specification for the server.

Parameters:
`statsSpec` - the statistics specification string.
Since:
7.1
See Also:
`StatsSpec`

---

getTraceSpec

`String` **`getTraceSpec`**`()`

Retrieve the current trace specification for the server.

Returns:
the trace specification string.
Since:
7.1

---

setTraceSpec

`void` **`setTraceSpec`**`(String traceSpec)`

Set the trace specification for the server.

Parameters:

`traceSpec` - the statistics specification string.

Since:

7.1

See Also:

`ObjectGridManager.setTraceSpecification(String)`

---

getEnvironmentInfo

`CompositeData` **`getEnvironmentInfo`**`()`

Retrieve the environment information for the server (host name, WebSphere eXtreme Scale version, and additional information). The CompositeData contains the following items:

Item Name	Type	Description
JMXServicePort	String	JMX Service Port
WASServerName	String	WebSphere Application Server Full Server Name
JVMVersion	String	JAVA Version
JMXConnectorPort	String	JMX Connector Port
IPAddress	String	IP Address
JavaVM	String	JVM Version
WASInstallRoot	String	WebSphere Application Server Product Directory
OSGIFrameworkVersion	String	OSGi Version
ClientPort	String	Client Port
HostName	String	Host name
Timestamp	String	Time Stamp from Server
WASBaseVersion	String	IBM WebSphere Application Server Version
OSName	String	Operating System
XSInstallRoot	String	WebSphere eXtreme Scale Product Directory
OSArch	String	OS Architecture
PeerPort	String	Peer Port
ServerType	String	Server Type
HAManagerPort	String	HAManager Port
JVMInstallPath	String	JAVA Directory
XC10Model	String	Machine Type and Model
JavaRuntimeInfo	String	JVM Runtime Version
xioContainerTCPNonSecure	String	XIO TCP/IP Port
JMXServiceURL	String	JMX Service Port
listenerPort	String	Listener Port (ORB)
ServerName	String	Server Name
WASXDVersion	String	IBM WebSphere Application Server - XD Version
WASExpressVersion	String	IBM WebSphere Application Server - ND Version
JavaBitMode	String	JAVA Bit Mode
XSVersion	String	WebSphere eXtreme Scale Version
JVMVendor	String	JAVA Vendor
OSVersion	String	Operating System Version
xioContainerTCPSecure	String	XIO TCP/IP SSL Port
PID	String	Process ID
WASNDVersion	String	IBM WebSphere Application Server - ND Version
ORB Version	String	ORB Version

Returns:

CompositeData containing the environment information

Since:
8.5, XC10 2.5

---

setLogNotificationFilter

`void` **`setLogNotificationFilter`**(`String` `regExFilter`)

Sets the regular expression filter to be applied in screening log record messages before being broadcasted by the grid server as a JMX Notification of type:

`com.ibm.websphere.objectgrid.log`

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The logging level.	8.6
LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The class name.	8.6
SourceMethodName	String	The method name.	8.6
SequenceNumber	Long	The sequence number	8.6
ThreadID	Integer	The thread id.	8.6
ThrownMessage	String	The thrown exception message or empty string, if not available.	8.6

Message: The log message.

Since: 8.6

Parameters:

`regExFilter` - The regular expression filter to applied in screening log record messages.

Since:
8.6, XC10 2.5

---

getLogNotificationFilter

`String` **`getLogNotificationFilter`**()

Retrieves current regular expression filter being applied while screening log record messages before being broadcasted as JMX notifications by the grid server as a JMX Notification of type:

`com.ibm.websphere.objectgrid.log`

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The logging level.	8.6

LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The class name.	8.6
SourceMethodName	String	The method name.	8.6
SequenceNumber	Long	The sequence number	8.6
ThreadID	Integer	The thread id.	8.6
ThrownMessage	String	The thrown exception message or empty string, if not available.	8.6

Message: The log message.

Since: 8.6

Returns:
Current regular expression filter being applied in screening log record messages.

Since:
8.6, XC10 2.5

---

checkFFDCNotification

`void` **`checkFFDCNotification`**`()`

Triggers a simulated exception to be captured as a first-failure data capture (FFDC) event (and subsequently broadcasted as a JMX notification), as a means to test and verify the monitoring being enabled on the server. The JMX notification generated and broadcasted has the following type and content:

`com.ibm.websphere.objectgrid.ffdc`

Description: All first-failure data captured by the grid server.

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
SourceID	String	The source code generating the exception.	8.6
ProbeID	String	The source code location associated with the exception captured.	8.6
ExceptionName	String	The exception captured.	8.6
Count	Integer	The current occurrence count for the specified exception.	8.6
DateOfFirstOccurrence	String	The exception's first occurrence time stamp.	8.6
Label	String	The label associated with the captured exception occurrence.	8.6

Message: A simulated first-failure data capture (FFDC) exception notification was generated by the server.

Since: 8.6

Since:
8.6, XC10 2.5

checkLoggingNotification

`void` **`checkLoggingNotification`**`()`

Generates a set of log records with various severity levels, providing a simple means to test and verify when the JMX notification monitoring being enabled on the grid server. Each of the JMX notifications generated and broadcasted has the following type and content:

`com.ibm.websphere.objectgrid.log`

UserData: A CompositeData with the following items:

Item Name	Type	Description	Since
LevelName	String	The logging level.	8.6
LoggerName	String	The name of the logger.	8.6
SourceClassName	String	The class name.	8.6
SourceMethodName	String	The method name.	8.6
SequenceNumber	Long	The sequence number	8.6
ThreadID	Integer	The thread id.	8.6
ThrownMessage	String	A simulated log info/warning/error notification was generated by the server	8.6

Message: A simulated log info/warning/error notification was generated by the server.

Since: 8.6

Since:

8.6, XC10 2.5

---

getRevisions

`TabularData` **`getRevisions`**`()`

Provides revisions for all shards in the server in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Grid	String	The object grid name.	8.6
MapSet	String	The name of the mapSet.	8.6
PartitionNumber	Integer	The number of a given partition.	8.6
ContainerName	String	Name of the container.	8.6
ShardType	String	Type of the shard.	8.6
RevisionState	TabularData	The lifetimeId / revisionNumber info for a given partition.	8.6

The RevisionState TabularData's CompositeData contains the following items:

Item Name	Type	Description	Since
LifetimeId	Long	The LifetimeId for the shard	8.6
Revision	Long	The revision number	8.6

Returns:
A TabularData object with the revision information.
Since:
8.6, XC10 2.5

---

getPrimaryRevisions

`TabularData` **`getPrimaryRevisions`**`()`

Provides revisions for all primary shards in the container in the form of a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Grid	String	The object grid name.	8.6
MapSet	String	The name of the mapSet.	8.6
PartitionNumber	Integer	The number of a given partition.	8.6
ContainerName	String	Name of the container.	8.6
ShardType	String	Type of the shard.	8.6
RevisionState	TabularData	The lifetimeId / revisionNumber info for a given partition.	8.6

The RevisionState TabularData's CompositeData contains the following items:

Item Name	Type	Description	Since
LifetimeId	Long	The LifetimeId for the shard	8.6
Revision	Long	The revision number	8.6

Returns:
A TabularData object with the revision information.
Since:
8.6, XC10 2.5

---

com.ibm.websphere.objectgrid.management

Interface ContainerMBean

public interface **ContainerMBean**

This MBean interface allows a client process to perform operations on and get status from an ObjectGrid container running in a dynamic environment. The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGridContainer,name=<server>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
WAS XD 6.1 FIX3, XC10

Field Summary

static String	**INVALID_PARTITION** INVALID_PARTITION indicates that no partition was found for the requested shard.
static String	**MAPSET_UNSUPPORTED_ON_CONTAINER** MAPSET_UNSUPPORTED_ON_CONTAINER indicates that an attempt was made to reserve a shard from a map set that is not supported on this container.
static String	**QUIESCE_COMPLETE** QUIESCE_COMPLETE is the MBean notification type for a completed quiesce.
s	

static String	**RELEASE_SUCCESSFUL** RELEASE_SUCCESSFUL indicates that the attempt to release the shard was successful.
static String	**RELEASE_UNSUPPORTED_WITH_PER_CONTAINER** RELEASE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy.
static String	**RESERVATION_PRIOR_TO_INITIAL_PLACEMENT** RESERVATION_PRIOR_TO_INITIAL_PLACEMENT indicates that the attempt to reserve the shard was processed successfully.
static String	**RESERVATION_SUCCESSFUL** RESERVATION_SUCCESSFUL indicates that the attempt to reserve the shard was successful.
static String	**RESERVE_UNSUPPORTED_WITH_PER_CONTAINER** RESERVE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy.
static String	**SHARD_ALREADY_RESERVED** SHARD_ALREADY_RESERVED indicates that the shard is already reserved elsewhere and cannot be reserved on the specified container.
sta	

static String	**SHARD_NOT_RESERVED_ON_CONTAINER** SHARD_NOT_RESERVED_ON_CONTAINER indicates that the attempt to release the shard from the requesting container failed because the specified shard was not found to be reserved by the requesting container.

Method Summary

int	**getActivatedShardCount**() Retrieve the total number of shards that have been activated for the life of this ObjectGrid container.
int	**getActiveShardCount**() Retrieve the number of active shards hosted in this ObjectGrid container.
String	**getContainerName**() Retrieve the name of the container.
int	**getDeactivatedShardCount**() Retrieve the total number of shards that have been deactivated for the life of this ObjectGrid container.
String	**getDomainName**() Retrieve the name of the catalog server grouping administering this container.
String	**getStatus**() Retrieve the status information for the shards in this container.
String	**getZoneName**() Retrieve the name of the zone grouping that this container belongs to.
int	**quiesceContainer**(Boolean inQuiesce) Prepare the container for a potential shutdown by moving replica shards, verifying that primaries have required sync replicas and preventing the placement of new shards.
String	**release**(String objectGridName, String mapSetName, String partitionName) Release a shard that has been previously reserved by this container.
String	**reserve**(String objectGridName, String mapSetName, String partitionName, String shardType) Reserve a specific shard on this container.

String	**retrieveStatus**(String objectGridName, String mapSetName) Retrieve the status information for the shards in this container, filtered by ObjectGrid and/or mapset.
void	**teardown**() Tears down and stops the container in a way to allow partitions to be moved to new locations.
void	**terminate**() Terminates a container without coordinating partition movement, partitions will failover.

Field Detail

QUIESCE_COMPLETE

static final String **QUIESCE_COMPLETE**

QUIESCE_COMPLETE is the MBean notification type for a completed quiesce.

See Also:
quiesceContainer(Boolean), Constant Field Values

RESERVATION_SUCCESSFUL

static final String **RESERVATION_SUCCESSFUL**

RESERVATION_SUCCESSFUL indicates that the attempt to reserve the shard was successful. The shard is reserved by the requesting container.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

RESERVATION_PRIOR_TO_INITIAL_PLACEMENT

static final String **RESERVATION_PRIOR_TO_INITIAL_PLACEMENT**

RESERVATION_PRIOR_TO_INITIAL_PLACEMENT indicates that the attempt to reserve the shard was processed successfully. However, since initial placement has not yet occurred, the reserved shard is not immediately moved to the requesting container. The shard will be placed on the container when initial placement is triggered.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

SHARD_ALREADY_RESERVED

static final String **SHARD_ALREADY_RESERVED**

SHARD_ALREADY_RESERVED indicates that the shard is already reserved elsewhere

and cannot be reserved on the specified container. The shard must be released from the owning container before it can be reserved again.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

INVALID_PARTITION

static final String **INVALID_PARTITION**

INVALID_PARTITION indicates that no partition was found for the requested shard.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

RESERVE_UNSUPPORTED_WITH_PER_CONTAINER

static final String **RESERVE_UNSUPPORTED_WITH_PER_CONTAINER**

RESERVE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy. Shard reservation is not supported with this placement strategy.

Since:
7.0.0.0 FIX1
See Also:
reserve(String, String, String, String), Constant Field Values

---

RELEASE_SUCCESSFUL

static final String **RELEASE_SUCCESSFUL**

RELEASE_SUCCESSFUL indicates that the attempt to release the shard was successful. The shard is no longer reserved by this container. The shard is free to migrate, but it is not forced to migrate.

Since:
7.0.0.0 FIX1
See Also:
release(String, String, String), Constant Field Values

SHARD_NOT_RESERVED_ON_CONTAINER

static final String **SHARD_NOT_RESERVED_ON_CONTAINER**

SHARD_NOT_RESERVED_ON_CONTAINER indicates that the attempt to release the shard from the requesting container failed because the specified shard was not found to be reserved by the requesting container. Only the container owning the reservation may release a shard.

Since:
7.0.0.0 FIX1
See Also:
release(String, String, String), Constant Field Values

RELEASE_UNSUPPORTED_WITH_PER_CONTAINER

`static final String` **`RELEASE_UNSUPPORTED_WITH_PER_CONTAINER`**

RELEASE_UNSUPPORTED_WITH_PER_CONTAINER indicates that the shard is part of a map set using the PER_CONTAINER placement strategy. Shard release is not supported with this placement strategy.

Since:
7.0.0.0 FIX1

See Also:
`release(String, String, String)`, Constant Field Values

MAPSET_UNSUPPORTED_ON_CONTAINER

`static final String` **`MAPSET_UNSUPPORTED_ON_CONTAINER`**

MAPSET_UNSUPPORTED_ON_CONTAINER indicates that an attempt was made to reserve a shard from a map set that is not supported on this container. Only map sets that were included in the deployment policy at container initialization are supported to run on this container.

Since:
7.1

See Also:
`reserve(String, String, String, String)`, Constant Field Values

Method Detail

teardown

`void` **`teardown`**`()`

Tears down and stops the container in a way to allow partitions to be moved to new locations.

---

terminate

`void` **`terminate`**`()`

Terminates a container without coordinating partition movement, partitions will failover.

---

getActiveShardCount

`int` **`getActiveShardCount`**`()`

Retrieve the number of active shards hosted in this ObjectGrid container.

Returns:
The current number of active shards.

---

getActivatedShardCount

`int` **`getActivatedShardCount`**`()`

Retrieve the total number of shards that have been activated for the life of this

ObjectGrid container.

Returns:
The number of activated shards.

---

getDeactivatedShardCount

`int` **`getDeactivatedShardCount`**`()`

Retrieve the total number of shards that have been deactivated for the life of this ObjectGrid container.

Returns:
The number of deactivated shards

---

getDomainName

`String` **`getDomainName`**`()`

Retrieve the name of the catalog server grouping administering this container.

Returns:
The domain name.

---

getZoneName

`String` **`getZoneName`**`()`

Retrieve the name of the zone grouping that this container belongs to.

Returns:
The name of the zone.

---

quiesceContainer

`int` **`quiesceContainer`**`(Boolean inQuiesce)`

Prepare the container for a potential shutdown by moving replica shards, verifying that primaries have required sync replicas and preventing the placement of new shards.

Parameters:
`inQuiesce` - Initiate quiesce mode (true) or cancel quiesce mode (false)
Returns:
The number of replicas moved off of the ObjectGrid container
See Also:
`QUIESCE_COMPLETE`

---

getStatus

`String` **`getStatus`**`()`

Retrieve the status information for the shards in this container.

Returns:
The status information for the shards in this container.

---

retrieveStatus

```
String retrieveStatus(String objectGridName,
                      String mapSetName)
```

Retrieve the status information for the shards in this container, filtered by ObjectGrid and/or mapset. For example, calling retrieveStatus with "og1" and "ms1" as parameters will return the partition status for those partitions in ObjectGrid og1 and mapset ms1. Passing in an empty string ("") objectGridName or mapSetName will return all of the partitions, since the empty string acts as a wildcard. Passing in the empty string for both parameters will return the same status as calling getStatus().

Parameters:

`objectGridName` - The name of the ObjectGrid for which the status is requested.
`mapSetName` - The name of the mapset within the ObjectGrid for which the status is requested.

Returns:

The status information for the shards in this container.

---

reserve

```
String reserve(String objectGridName,
               String mapSetName,
               String partitionName,
               String shardType)
```

Reserve a specific shard on this container. Calling this method will cause the requested shard to move to this container. The shard can be moved to this container only if it is not reserved elsewhere. Calling this method prior to initial placement will pre-reserve the shard so that it will be placed onto this container when initial placement occurs. If non-reserved shard for the same partition is on this container prior to reservation, the non-reserved shard will be moved off the container upon reservation. A reserved shard will not be moved off of this container until it is released or the container is stopped.

Parameters:

`objectGridName` - the ObjectGrid containing the shard
`mapSetName` - the map set containing the shard
`partitionName` - the partition containing the shard
`shardType` - the type of shard. Currently, only primary shards can be reserved: `ShardMBean.TYPE_PRIMARY`

Returns:

the return code indicating the result of the reserve request

Throws:

`IllegalArgumentException` - if any of the arguments are null or the empty String. Also thrown if shardType is not `ShardMBean.TYPE_PRIMARY`

Since:

7.0.0.0 FIX1

See Also:

`ShardMBean.TYPE_PRIMARY`, `release(String, String, String)`, `RESERVATION_SUCCESSFUL`, `RESERVATION_PRIOR_TO_INITIAL_PLACEMENT`, `SHARD_ALREADY_RESERVED`, `INVALID_PARTITION`, `RESERVE_UNSUPPORTED_WITH_PER_CONTAINER`

---

release

```
String release(String objectGridName,
               String mapSetName,
               String partitionName)
```

Release a shard that has been previously reserved by this container. This container can only release shards that it has reserved. Releasing the shard does not guarantee the

shard will be moved. The shard may remain on this container. However, it will not be explicitly bound to this container. Releasing a shard allows the shard to move freely to other containers or to be reserved by another container.

Parameters:

`objectGridName` - the ObjectGrid containing the shard
`mapSetName` - the map set containing the shard
`partitionName` - the partition containing the shard

Returns:

the return code indicating the result of the release request

Throws:

`IllegalArgumentException` - if any of the arguments are null or the empty String

Since:

7.0.0.0 FIX1

See Also:

`reserve(String, String, String, String)`, `RELEASE_SUCCESSFUL`, `SHARD_NOT_RESERVED_ON_CONTAINER`, `RELEASE_UNSUPPORTED_WITH_PER_CONTAINER`

---

getContainerName

`String` **`getContainerName`**`()`

Retrieve the name of the container. The container name is based on the server name and includes a suffix which uniquely identifies the container within the server.

Returns:

the name of the container

Since:

7.1

---

com.ibm.websphere.objectgrid.management

Interface CatalogServiceManagementMBean

public interface **CatalogServiceManagementMBean**

This MBean interface allows user to manipulate the behaviors of heartbeat and leader manager and other catalog service specific actions. The object name pattern for this MBean is:

```
com.ibm.websphere.objectgrid:type=CatalogService
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:
7.1, XC10

Field Summary

static String	**COREGROUP_MEMBERSHIP_CHANGE** Constant representing a core group membership change notification.
static int	**HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE** Constant representing a heartbeat frequency level at an aggressive rate.
static int	**HEARTBEAT_FREQUENCY_LEVEL_RELAXED** Constant representing a heartbeat frequency level at relaxed rate.
static int	**HEARTBEAT_FREQUENCY_LEVEL_TYPICAL** Constant representing a heartbeat frequency level at a typical rate.

int	
static String	**MANAGEMENT_CONCENTRATOR_STATUS_STARTED**
static String	**MANAGEMENT_CONCENTRATOR_STATUS_STOPPED**
static String	**ORB** Constant representing the ORB or Object Request Broker transport
static String	**SERVER_EVENT_STARTED** Constant representing an eXtreme Scale server start notification.
static String	**SERVER_EVENT_STOPPED** Constant representing an eXtreme Scale server stop notification.
static String	**XIO** Constant representing the eXtremeIO transport

[n g](#)	

Method Summary

[T a b u l a r D a t a](#)	**getContainerReplicationState**() Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross containers within one domain.
[T a b u l a r D a t a](#)	**getDomainReplicationState**() Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross domains.
int	**getHeartBeatFrequencyLevel**() Retrieves the heartbeat frequency level.
[S t r i n g](#)	**getManagementConcentratorStatus**() Returns the String representation of the management concentrator status.
int	**getNumberOfServers**() Retrieves the number eXtreme Scale servers that are currently registered with the catalog service.
[C o m p o s i t e D a t a](#)	**getServers**() Retrieves a CompositeData of each eXtreme Scale server that is currently registered with the catalog service.
[S t r i n g](#)	**getTransport**() Returns the transport used by the catalog service domain.
boolean	**isPrimary**() Provides indication if the catalog is primary.

void	**logMessage**(String level, String message) Provides support for logging user messages from processes outside the catalog and/or container servers.
void	**startManagementConcentrator**() Starts the Management Concentrator.
void	**stopManagementConcentrator**() Stops the Management Concentrator.

Field Detail

MANAGEMENT_CONCENTRATOR_STATUS_STARTED

static final String **MANAGEMENT_CONCENTRATOR_STATUS_STARTED**

See Also:
Constant Field Values

MANAGEMENT_CONCENTRATOR_STATUS_STOPPED

static final String **MANAGEMENT_CONCENTRATOR_STATUS_STOPPED**

See Also:
Constant Field Values

COREGROUP_MEMBERSHIP_CHANGE

static final String **COREGROUP_MEMBERSHIP_CHANGE**

Constant representing a core group membership change notification. The user data associated with this notification is a CompositeData.

The CompositeData includes the following items:

Item Name	Type	Description
MemberName	String	The name of the server that is included in the core group.

See Also:
Constant Field Values

SERVER_EVENT_STARTED

static final String **SERVER_EVENT_STARTED**

Constant representing an eXtreme Scale server start notification.

The UserData argument of the Notification includes a TabularData that includes information for each of the servers. Each CompositeData (row in the TabularData) contains the following items:

	Description

Item Name	Type	
HAPort	String	The port number of the high availability manager.
Host	String	The host/ip address of the server.
JMXServiceURL	String	The JMX service url used to access the server.
ServerName	String	The name of the server.
ZoneName	String	The name of the zone that the server belongs.

See Also:
Constant Field Values

SERVER_EVENT_STOPPED

`static final String` **`SERVER_EVENT_STOPPED`**

Constant representing an eXtreme Scale server stop notification.

The UserData argument of the Notification includes a TabularData instance where each CompositeData contains the following items:

Item Name	Type	Description
ServerName	String	The name of the server.

See Also:
Constant Field Values

HEARTBEAT_FREQUENCY_LEVEL_TYPICAL

`static final int` **`HEARTBEAT_FREQUENCY_LEVEL_TYPICAL`**

Constant representing a heartbeat frequency level at a typical rate.

A typical heartbeat frequency allows reasonable failover detection and resource utilization. This value is the default.

See Also:
Constant Field Values

HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE

`static final int` **`HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE`**

Constant representing a heartbeat frequency level at an aggressive rate.

An increased heartbeat frequency allows failures to be detected more quickly, but can also uses additional CPU and network resources. This level is more sensitive to missing heartbeats when the server is stressed.

See Also:
Constant Field Values

HEARTBEAT_FREQUENCY_LEVEL_RELAXED

`static final int` **`HEARTBEAT_FREQUENCY_LEVEL_RELAXED`**

Constant representing a heartbeat frequency level at relaxed rate.

A decreased heartbeat frequency increases the time to detect failures, but also decreases

CPU and network utilization.

See Also:
Constant Field Values

ORB

```
static final String ORB
```

Constant representing the ORB or Object Request Broker transport

See Also:
Constant Field Values

XIO

```
static final String XIO
```

Constant representing the eXtremeIO transport

See Also:
Constant Field Values

Method Detail

getHeartBeatFrequencyLevel

```
int getHeartBeatFrequencyLevel()
```

Retrieves the heartbeat frequency level.

Valid values include:

- HEARTBEAT_FREQUENCY_LEVEL_TYPICAL
- HEARTBEAT_FREQUENCY_LEVEL_RELAXED
- HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE

Returns:
the heartbeat frequency level: -1, 0 or 1 as defined by the constants that begin with name HEARTBEAT_FREQUENCY_LEVEL.

---

getServers

```
CompositeData getServers()
```

Retrieves a CompositeData of each eXtreme Scale server that is currently registered with the catalog service.

The CompositeData includes the following items:

Item Name	Type	Description
serverName	String	The name of the server that is registered with the catalog service.

Returns:
the CompositeData representing the currently registered eXtreme Scale servers.

---

getNumberOfServers

`int` **`getNumberOfServers`**`()`

Retrieves the number eXtreme Scale servers that are currently registered with the catalog service.

Returns:
the number of registered eXtreme Scale servers.

---

logMessage

`void` **`logMessage`**`(String level,`
`String message)`

Provides support for logging user messages from processes outside the catalog and/or container servers. Example: XC10 surfaced SNMP trap messages can be flowed from the SNMP agent which throws traps in the console server (sMash) process, not typically an XS catalog/container server.

Parameters:
`level` - name describing the severity of the event which is compatible with java.util.logging.Level.parse(String name) where name may be either level name (ex. "SEVERE") or an integer value (ex. "1000") - @see `Level.parse(String)`
`message` - for the end user (already sNLS rendered)

Since:
8.6, XC10 2.5

---

isPrimary

`boolean` **`isPrimary`**`()`

Provides indication if the catalog is primary.

Returns:
true for primary catalog.

Since:
8.6, XC10 2.5

---

getContainerReplicationState

`TabularData` **`getContainerReplicationState`**`()`

Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross containers within one domain. For a given container, outstanding out-bound revisions need to be replicated from primary shards located in this container into replicas located in other containers. In similar way, outstanding in-bound revisions need to be replicated from primary shards located in other containers into corresponding replicas located in this container. This operation can be used to check the differences in data revisions between containers within one domain.

The result is a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Container	String	The container ID.	8.6
OutboundRevisions	Long	Number of out-bound revisions for container.	8.6
InboundRevisions	Long	Number of in-bound revisions for container.	8.6

Returns:

A TabularData object with the container replication state information.

Since:

8.6, XC10 2.5

---

getDomainReplicationState

`TabularData` **`getDomainReplicationState`**`()`

Provides the numbers of outstanding in-bound and out-bound revisions which need to be replicated cross domains. Outstanding out-bound revisions need to be replicated from local primary shards into corresponding remote primary shards. Outstanding in-bound revisions need to be replicated from remote primary shards into corresponding local primary shards. This operation can be used to check the differences in data revisions between different domains linked by MMR replication.

The result is a TabularData object, where each CompositeData (row in the TabularData) contains the following items:

Item Name	Type	Description	Since
Domain	String	The domain name.	8.6
Container	String	The container ID.	8.6
OutboundRevisions	Long	Number of out-bound revisions for container.	8.6
InboundRevisions	Long	Number of in-bound revisions for container.	8.6

Returns:

A TabularData object with the domain replication state information.

Since:

8.6, XC10 2.5

---

getTransport

`String` **`getTransport`**`()`

Returns the transport used by the catalog service domain.

Returns:

String containing the transport type

Since:

8.6, XC10 2.5

See Also:

`ORB`, `XIO`

---

startManagementConcentrator

`void` **`startManagementConcentrator`**`()`

Starts the Management Concentrator. The catalog server will now start listening for log messages.

Since:

8.6.0.2, XC10 2.5

---

stopManagementConcentrator

`void` **`stopManagementConcentrator`**`()`

Stops the Management Concentrator. The catalog server will no longer listen for log

messages. Stopping the Management Concentrator will also stop the Message Center in the web monitoring console.

Since:
8.6.0.2, XC10 2.5

---

getManagementConcentratorStatus

`String` **`getManagementConcentratorStatus()`**

Returns the String representation of the management concentrator status. Status will be either CatalogServiceManagementMBean.MANAGEMENT_CONCENTRATOR_STATUS_STARTED or CatalogServiceManagementMBeanMANAGEMENT_CONCENTRATOR_STATUS_STOPPED.

Returns:
String containing the status

Since:
8.6.0.2, XC10 2.5

---

com.ibm.websphere.objectgrid.management

Interface AgentManagerMBean

public interface **AgentManagerMBean**

This MBean interface allows a client process to access different attributes and statistical data about a specific Agent on a server process. The Agent Manager MBean is scoped at the map level and therefore can access statistical data for every agent run against the specified map. In a dynamic ObjectGrid environment, the object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=AgentManager,name=Agent-<map>,partition=<partition id>,objectgr
id=<objectgrid>,host=<host>,ogServerName=<server>
```

If ObjectGrid is running in a WebSphere Application Server process, more key=value pairs may be added to the object name.

Since:

WAS XD 6.1.0.5, XC10

Method Summary

double	**getAgentInflationTime**(String agentClassName) Gets the specified agent's inflation time attribute loaded up by the `retrieveStatsModule()` method.
double	**getAgentSerializationTime**(String agentClassName) Gets the specified agent's serialization time attribute loaded up by the `retrieveStatsModule()` method.
double	**getFailureCount**(String agentClassName) Gets the specified agent's failure count attribute loaded up by the `retrieveStatsModule()` method.
double	**getInvocationCount**(String agentClassName) Gets the specified agent's invocation count attribute loaded up by the `retrieveStatsModule()` method.
double	**getPartitionCount**(String agentClassName) Gets the specified agent's partition count attribute loaded up by the `retrieveStatsModule()` method.

double	**getReduceTime**(String agentClassName) Gets the specified agent's total reduce time attribute loaded up by the `retrieveStatsModule()` method.
double	**getResultInflationTime**(String agentClassName) Gets the specified agent's result inflation time attribute loaded up by the `retrieveStatsModule()` method.
double	**getResultSerializationTime**(String agentClassName) Gets the specified agent's result serialization time attribute loaded up by the `retrieveStatsModule()` method.
double	**getTotalDurationTime**(String agentClassName) Gets the specified agent's total run time attribute loaded up by the `retrieveStatsModule()` method.
com.ibm.websphere.objectgrid.stats.AgentStatsMo	**retrieveStatsModule**(String agentClassName) Gets the `AgentStatsModule` used to retrieve statistics associated with the specified agent class

d u l e	

Method Detail

getReduceTime

`double` **`getReduceTime`**(`String` `agentClassName`)

Gets the specified agent's total reduce time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the reduce time for this agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getReduceTime(boolean copy)`

---

getTotalDurationTime

`double` **`getTotalDurationTime`**(`String` `agentClassName`)

Gets the specified agent's total run time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the total run time for this agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getTotalDurationTime(boolean copy)`

---

getAgentSerializationTime

`double` **`getAgentSerializationTime`**(`String` `agentClassName`)

Gets the specified agent's serialization time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to serialize the agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getAgentSerializationTime(boolean copy)`

---

getAgentInflationTime

`double` **`getAgentInflationTime`**(`String` `agentClassName`)

Gets the specified agent's inflation time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to inflate the agent in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getAgentInflationTime(boolean copy)`

---

getResultInflationTime

`double` **`getResultInflationTime`**`(String agentClassName)`

Gets the specified agent's result inflation time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to inflate the agent results for a given partition in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getResultInflationTime(boolean copy)`

---

getResultSerializationTime

`double` **`getResultSerializationTime`**`(String agentClassName)`

Gets the specified agent's result serialization time attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the time it takes to serialize the agent results for a given partition in milliseconds

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getResultSerializationTime(boolean copy)`

---

getPartitionCount

`double` **`getPartitionCount`**`(String agentClassName)`

Gets the specified agent's partition count attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the number of partitions this agent is sent to

See Also:

`retrieveStatsModule(String)`, `AgentStatsModule.getPartitionCount(boolean copy)`

---

getFailureCount

`double` **`getFailureCount`**`(String agentClassName)`

Gets the specified agent's failure count attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the failure count for the specified agent

See Also:

retrieveStatsModule(String), AgentStatsModule.getFailureCount(boolean copy)

---

getInvocationCount

double **getInvocationCount**(String agentClassName)

Gets the specified agent's invocation count attribute loaded up by the `retrieveStatsModule()` method.

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

the invocation count for the specified agent

See Also:

retrieveStatsModule(String), AgentStatsModule.getInvocationCount(boolean copy)

---

retrieveStatsModule

com.ibm.websphere.objectgrid.stats.AgentStatsModule **retrieveStatsModule**(String agentClassName)

Gets the `AgentStatsModule` used to retrieve statistics associated with the specified agent class

Parameters:

`agentClassName` - The fully qualified class name of the agent

Returns:

an AgentStatsModule for statistics associated with the specified agent class

See Also:

`AgentStatsModule`

Package com.ibm.websphere.objectgrid.plugins

These are the interfaces for adding plugins to the Grid core framework.

See:

Description

Interface Summary

BeanFactory	Implement this interface to allow bean factories like Spring or Google guice to be integrated.
CacheEntry	This interface represents a cache entry in an ObjectGrid map.
EvictionEventCallback	An instance of `EvictionEventCallback` is passed into the `Evictor` at initialization time.
Evictor	Data contained in a `BackingMap` are evicted when the map is full.
EvictorData	This interface is optionally used by an implementator of the `Evictor` interface.
LogElement	LogElements are the individual entries within a LogSequence.
LogSequence	LogSequence is the ordered list of changes performed against a given map for a given transaction.
LogSequenceFilter	This interface can be used to filter a LogSequence.
MapEventListener	This callback interface is implemented by the application when it wants to receive events about a Map such as the eviction of a map entry.
ObjectGridEventGroup	This is a set of single method interfaces for fine grained events delivered for an ObjectGrid.
ObjectGridEventGroup.ShardEvents	These events are fired when a shard is made a primary shard and when the shard is demoted from a primary.
ObjectGridEventGroup.ShardLifecycle	These events are fired when an ObjectGrid shard is initialized and destroyed.
ObjectGridEventGroup.TransactionEvents	These events are called every single transaction.
ObjectGridEventListener	This interface is used to create an implementation of an event listener for an ObjectGrid.
ReplicationMapListener	**Deprecated.** *The client replicated map function is deprecated in version 8.6.*
Transaction	Calling methods on a `Session` will send corresponding events to the

Callback	TransactionCallback.
TransactionCallback.BeforeCommit	The BeforeCommit optional mix-in interface for the TransactionCallback plug-in interface allows plug-ins to be notified at the beginning of a `Session.commit()`.
TransactionCallback.BeforeCommit.TransactionContext	The TransactionContext identifies various information that's available to the beforeCommit() method.
ValueProxyInfo	This interface can be used by value objects to inform a BackingMap or Loader which attribute(s) have been dirtied.

Class Summary

EvictorData.SpecialEvictorData	Special value class used for representing the key not being found in the `BackingMap`.
LogElement.Type	The `Type` class is used to represent a `LogElement` type.

Exception Summary

CacheEntryException	This exception indicates an error occurred during a cache entry operation.
LoaderException	This exception is the base exception for any exceptions encountered by a `Loader`.
TransactionCallbackException	This exception is thrown when a method call to TransactionCallback fails.

Package com.ibm.websphere.objectgrid.plugins Description

These are the interfaces for adding plugins to the Grid core framework.

Overview

These plugins can be added into ObjectGrid in several ways such as xml configuration, programmatically adding, or using annotation.

Annotation based callbacks

ObjectGrid when running on Java 5 will start to use an annotated method callback system. This means that objects can be registered as callbacks or listeners. The methods on the object must be annotated as to be invoked for a certain event. Unannotated methods are not invoked. The name of the method is unimportant. The method arguments and return type must be the same as expected for the callback method.

Why?

Usually, callbacks are specified using an interface. This works well but results in a possible performance loss as all methods on the interface will be invoked by the ObjectGrid even

though the application is only interested in a single event. This wastes precious resources. Another issue is when we need to add a new event. Adding a new method to an existing interface breaks back wards compatibility. We can make a new interface extending the old one with the new methods but this is also undesirable as soon there are many interfaces in the hierarchy as new events are added. The annotation system allows the application to only mark methods to be called avoiding the first problem and if new event types are added they have no impact on existing callback objects. Newer applications can add a method and annotate it with the new event annotation to receive the event.

---

com.ibm.websphere.objectgrid.plugins

Interface ValueProxyInfo

public interface **ValueProxyInfo**

This interface can be used by value objects to inform a BackingMap or Loader which attribute(s) have been dirtied. This mechanism allows the BackingMap and Loader to interrogate the set of changed attributes in the value object instead of just assuming the whole value object has been updated. For this to be useful, the application must only use the getter and setter methods defined for the value object's interface.

Since:

WAS XD 6.0, XC10

See Also:

CopyMode.COPY_ON_WRITE, BackingMap.setCopyMode(CopyMode, Class), Loader.batchUpdate(TxID, LogSequence)

Method Summary

void	**ibmClearDirtyAttributes**() Clears the list of dirty attributes.
List	**ibmGetDirtyAttributes**() Returns a list of dirty attributes based on the value interface set on the map.
Object	**ibmGetRealValue**() Returns the real value object this proxy represents.

Method Detail

ibmGetDirtyAttributes

List **ibmGetDirtyAttributes**()

Returns a list of dirty attributes based on the value interface set on the map.

The attribute name is always starts with an upper case letter. For example, if the setter for the attribute is setPrice then 'Price' is the string returned here. The runtime uses `substring(3)` of the setter method name as the attribute name.

Returns:

List of attribute names (Strings)

ibmGetRealValue

`Object` **`ibmGetRealValue`**`()`

Returns the real value object this proxy represents.

Needed internally by the BackingMap to return a separate proxy for each transaction.

Returns:
actual value object.

---

ibmClearDirtyAttributes

`void` **`ibmClearDirtyAttributes`**`()`

Clears the list of dirty attributes.

---

com.ibm.websphere.objectgrid.plugins

Class TransactionCallbackException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.TransactionCallbackException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException, ReplicationVotedToRollbackTransactionException

---

```
public class TransactionCallbackException
extends ObjectGridException
```

This exception is thrown when a method call to TransactionCallback fails.

Since:

WAS XD 6.0, XC10

See Also:

TransactionCallback, Serialized Form

Constructor Summary

TransactionCallbackException() Constructs a new TransactionCallbackException with `null` as its detail message.
TransactionCallbackException(String message) Constructs a new TransactionCallbackException with the specified detail message.
TransactionCallbackException(String message, Throwable cause) Constructs a new TransactionCallbackException with the specified detail message and cause.
TransactionCallbackException(Throwable cause) Constructs a new TransactionCallbackException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionCallbackException

`public` **`TransactionCallbackException`**`()`

Constructs a new TransactionCallbackException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

TransactionCallbackException

`public` **`TransactionCallbackException`**`(String message)`

Constructs a new TransactionCallbackException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

TransactionCallbackException

`public` **`TransactionCallbackException`**`(String message,`
`Throwable cause)`

Constructs a new TransactionCallbackException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionCallbackException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

TransactionCallbackException

`public` **`TransactionCallbackException`**`(Throwable cause)`

Constructs a new TransactionCallbackException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains

the class and detail message of `cause`). This constructor is useful for TransactionCallbackExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid.plugins

Interface TransactionCallback

All Known Subinterfaces:

TransactionCallback.BeforeCommit

All Known Implementing Classes:

WebSphereTransactionCallback

`public interface` **`TransactionCallback`**

Calling methods on a `Session` will send corresponding events to the TransactionCallback. An ObjectGrid can have zero or one TransactionCallback. BackingMaps defined on an ObjectGrid with a TransactionCallback should have corresponding Loaders.

A TransactionCallback works with `Loader`s and place transaction specific objects in slots on the `TxID` object that `Loader`s can obtain. Examples are database connections, prepared statement caches, etc. The TransactionCallback should reserve slots in the `TxID` by calling `ObjectGrid.reserveSlot(String)` using the name `TxID.SLOT_NAME`. The TransactionCallback can then put an object at that index in the `TxID`. A `Loader` can retrieve the index used by the `TransactionCallback` by calling an internal method on the `TransactionCallback`'s implementation. A reference to the configured `TransactionCallback` can be found using the `TxID.getSession().getObjectGrid().getTransactionCallback()` code sequence.

A `TransactionCallback` implementation that also implements the `ObjectGridLifecycleListener` interface will be automatically added as an `EventListener` on the `ObjectGrid` when the callback is set on the object grid.

A TransactionCallback may implement the `ObjectGridPlugin` interface in order to receive enhanced ObjectGrid plug-in lifecycle method calls. The plug-in is also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Since:

WAS XD 6.0, XC10

See Also:

`Loader`, `ObjectGrid.addEventListener(EventListener)`, `ObjectGrid.getTransactionCallback()`, `ObjectGrid.reserveSlot(String)`, `ObjectGrid.setTransactionCallback(TransactionCallback)`, `Session.getObjectGrid()`, `TxID.putSlot(int, Object)`, `TxID.getSlot(int)`, `TxID.getSession()`

Nested Class Summary

static interface	**TransactionCallback.BeforeCommit** The BeforeCommit optional mix-in interface for the TransactionCallback plug-in interface allows plug-ins to be notified at the beginning of a `Session.commit()`.

face

Method Summary

void	**begin**(TxID id) Invoked when starting a Session transaction.
void	**commit**(TxID id) Invoked when committing a Session transaction.
void	**initialize**(ObjectGrid objectGrid) Invoked when an `ObjectGrid` is initialized.
boolean	**isExternalTransactionActive**(Session session) Called when an application attempts to use a Session with no transaction active.
void	**rollback**(TxID id) Invoked when rolling back a Session transaction.

Method Detail

initialize

```
void initialize(ObjectGrid objectGrid)
                throws TransactionCallbackException
```

Invoked when an `ObjectGrid` is initialized.

This method is called so this object can do any implementation specific intialization.

Parameters:

`objectGrid` - A reference to the ObjectGrid.

Throws:

TransactionCallbackException - if an error occurs during processing

See Also:

ObjectGrid.reserveSlot(String)

---

begin

```
void begin(TxID id)
           throws TransactionCallbackException
```

Invoked when starting a Session transaction.

A TransactionCallback can communicate the begin processing (along with the TxID) to the appropriate BackingMap and/or Loader. The Loader may use this signal to start a

corresponding transaction on the underlying connection to a database.

Parameters:
`id` - transaction identifer (TxID)
Throws:
`TransactionCallbackException` - if an error occurs during processing
See Also:
`Session.begin()`, `Session.beginNoWriteThrough()`, `TxID`

---

commit

```
void commit(TxID id)
            throws TransactionCallbackException
```

Invoked when committing a Session transaction.

This method should be used to commit any underlying transaction and return any underlying connection back to the pool. The TxID is provided to determine which transaction is being committed

Parameters:
`id` - transaction identifier (TxID)
Throws:
`TransactionCallbackException` - if an error occurs during processing
See Also:
`begin(TxID)`, `Session.commit()`, `TxID`

---

rollback

```
void rollback(TxID id)
              throws TransactionCallbackException
```

Invoked when rolling back a Session transaction.

This method should be used to roll back any underlying transaction and return any underlying connection back to the pool. The TxID is provided to determine which transaction is being committed

Parameters:
`id` - transaction identifier (TxID)
Throws:
`TransactionCallbackException` - if an error occurs during processing
See Also:
`begin(TxID)`, `Session.rollback()`, `TxID`

---

isExternalTransactionActive

```
boolean isExternalTransactionActive(Session session)
```

Called when an application attempts to use a Session with no transaction active.

The callback could return `true` in which case an auto `Session.begin()` is executed. If `false` is returned, an application exception is thrown indicating no transaction is active. This event is usually used when integrating with a J2EE environment such as WebSphere Application Server.

Parameters:
`session` - the session which the application is using
Returns:

`true` if an auto begin should be done, `false` if this is not the case

See Also:

Session

---

com.ibm.websphere.objectgrid.plugins

Interface TransactionCallback.BeforeCommit

All Superinterfaces:
TransactionCallback

Enclosing interface:
TransactionCallback

```
public static interface TransactionCallback.BeforeCommit
extends TransactionCallback
```

The BeforeCommit optional mix-in interface for the TransactionCallback plug-in interface allows plug-ins to be notified at the beginning of a `Session.commit()`. Implementations can use the `beforeCommit()` method to validate changed data in the transaction and modify the data.

Since:
7.1.1

Nested Class Summary

static interface	**TransactionCallback.BeforeCommit.TransactionContext** The TransactionContext identifies various information that's available to the beforeCommit() method.

Nested classes/interfaces inherited from interface com.ibm.websphere.objectgrid.plugins.TransactionCallback

TransactionCallback.BeforeCommit

Method Summary

void	**beforeCommit**(TransactionCallback.BeforeCommit.TransactionContext ctx) Invoked at the beginning of a `Session.commit()`.

Methods inherited from interface com.ibm.websphere.objectgrid.plugins.TransactionCallback

begin, commit, initialize, isExternalTransactionActive, rollback

Method Detail

beforeCommit

```
void beforeCommit(TransactionCallback.BeforeCommit.TransactionContext ctx)
                  throws TransactionCallbackException
```

Invoked at the beginning of a `Session.commit()`.

Use the `TransactionContext.getLogSequences()` method to retrieve the changes made by this transaction. Use the `TransactionContext.getTxId().getSession()` methods to access the Session. The Session can be used to access ObjectMaps and modify data in the current transaction.

Parameters:

`ctx` - the context of the transaction.

Throws:

`TransactionCallbackException` - if an error occurs during processing. Any exception will roll back the transaction and will be included in the TransactionException thrown to the caller.

See Also:

`Session.commit()`, `TxID`

---

com.ibm.websphere.objectgrid.plugins

Interface TransactionCallback.BeforeCommit.TransactionContext

Enclosing interface:
TransactionCallback.BeforeCommit

```
public static interface TransactionCallback.BeforeCommit.TransactionContext
```

The TransactionContext identifies various information that's available to the beforeCommit() method.

Since:
7.1.1

Method Summary

Collection<LogSequence>	**getLogSequences**() The LogSequences that reflect the pending changes to the map.
TxID	**getTxID**() Retrieve the TxID for the transaction.

Method Detail

getTxID

TxID **getTxID**()

Retrieve the TxID for the transaction.

Returns:
the TxID for the transaction.

---

getLogSequences

`Collection<LogSequence> getLogSequences()`

The LogSequences that reflect the pending changes to the map.

Returns:
the LogSequences.

---

com.ibm.websphere.objectgrid.plugins

Interface ReplicationMapListener

Deprecated. *The client replicated map function is deprecated in version 8.6. Use the ContinuousQueryManager function.*

public interface **ReplicationMapListener**

This interface is used to create an implementation of an event listener for client-side maps that are in replication mode. Registered listeners receive notification callbacks for replication start and exit events and data changes.

Listener instances can be registered with a map using the ClientReplicableMap.enableClientReplication(com.ibm.websphere.objectgrid.ClientReplicableMap.Mode, int[], ReplicationMapListener) method.

Since:
WAS XD 6.1, XC10

Method Summary

void	**onData**(LogSequence logSequence) **Deprecated.** This method is invoked when a data change is replicated to the client replication map.
void	**replicationExits**() **Deprecated.** This method is invoked when replication has been disabled for this map.
void	**replicationStarts**() **Deprecated.** This method is invoked when a snapshot mode replica has been synchronized with the client or a continuous replica has started replicating.

Method Detail

replicationStarts

void **replicationStarts**()

Deprecated.
This method is invoked when a snapshot mode replica has been synchronized with the client or a continuous replica has started replicating.

See also:
ClientReplicableMap.enableClientReplication(com.ibm.websphere.objectgrid.ClientReplicableMap.Mode, int[], ReplicationMapListener)

onData

`void` **`onData`**(`LogSequence` `logSequence`)

Deprecated.
This method is invoked when a data change is replicated to the client replication map.

Parameters:

`logSequence` - the log sequence containing all of the data changes.

---

replicationExits

`void` **`replicationExits`**()

Deprecated.
This method is invoked when replication has been disabled for this map.

See Also:

`ClientReplicableMap.disableClientReplication()`

---

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventListener

All Superinterfaces:

com.ibm.websphere.objectgrid.plugins.EventListener

```
public interface ObjectGridEventListener
extends com.ibm.websphere.objectgrid.plugins.EventListener
```

This interface is used to create an implementation of an event listener for an ObjectGrid. Instances of `ObjectGridEventListener`s are set on the `ObjectGrid` interface. Any significant events are communicated to the application using the methods outlined below. When using Java 5, this callback also supports new callback annotation mechanism.

Since:

WAS XD 6.0, XC10

See Also:

`ObjectGrid.addEventListener(EventListener)`, `ObjectGrid.removeEventListener(EventListener)`, `EventListener`

Method Summary

void	**destroy**() Called when the `ObjectGrid` associated with this listener is destroyed.
void	**initialize**(Session session) Invoked when an `ObjectGrid` is initialized.
void	**transactionBegin**(String txid, boolean isWriteThroughEnabled) Signals the beginning of a `Session` transaction.
void	**transactionEnd**(String txid, boolean isWriteThroughEnabled, boolean committed, Collection changes) Signals the ending of a `Session` transaction.

Method Detail

initialize

```
void initialize(Session session)
```

Invoked when an `ObjectGrid` is initialized.

A usable `Session` instance is passed to this listener to provide all of the necessary access

to the various ObjectGrid objects.

Parameters:

`session` - a Session instance that this listener is associated with.

See Also:

`ObjectGrid.initialize()`

---

transactionBegin

```
void transactionBegin(String txid,
                      boolean isWriteThroughEnabled)
```

Signals the beginning of a `Session` transaction.

A stringified version of the TxID is provided for correlating with the end of the transaction, if so desired. The type of transaction is also provided by the isWriteThroughEnabled boolean parameter.

Parameters:

`txid` - Stringified version of the TxID

`isWriteThroughEnabled` - boolean flag indicating whether the Session transaction was started using the `Session.beginNoWriteThrough()`. method. `false` is passed if `beginNoWriteThrough()` was used.

See Also:

`Session.begin()`, `Session.beginNoWriteThrough()`

---

transactionEnd

```
void transactionEnd(String txid,
                    boolean isWriteThroughEnabled,
                    boolean committed,
                    Collection changes)
```

Signals the ending of a `Session` transaction.

A string version of the TxID is provided for correlating with the begin of the transaction, if so desired. Map changes are also reported with the collection of `LogSequence`s passed to this method. Typical uses of this event are for customers doing custom peer invalidation or peer commit push. This event listener gives them the changes. Calls to this method are made after commit and are sequenced so that they are delivered one by one, not in parallel. The event order is the commit and rollback order.

For an ObjectGridEventListener receiving changes in an `ObjectMap` that is configured to use a `OutputFormat.RAW` for the keys or values, the keys and values objects in the LogSequences will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the implementation class.

Parameters:

`txid` - string version of the TxID

`isWriteThroughEnabled` - boolean flag indicating whether the Session transaction was started using the `Session.beginNoWriteThrough()`. method. `false` is passed if `beginNoWriteThrough()` was used.

`committed` - a boolean flag indicating whether the transaction was committed (`true`) or rolled back (`false`)

`changes` - a Collection of LogSequences representing the changes that were committed or rolled back.

See Also:

LogSequence.isRollback(), Session.begin(), Session.beginNoWriteThrough(), Session.commit(), Session.rollback()

---

destroy

```
void destroy()
```

Called when the ObjectGrid associated with this listener is destroyed.

This method is the opposite of the initialize method. When it is called, the listener can free up any resources it uses.

See Also:

ObjectGrid.destroy()

---

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup

public interface **ObjectGridEventGroup**

This is a set of single method interfaces for fine grained events delivered for an ObjectGrid. Classes implementing these interfaces AND ObjectGridEventListener can receive these events. If an ObjectGridEventListener implements ANY of these interfaces that only the specific methods on the interfaces implemented will be called.

Since:

WAS XD 6.1, XC10

See Also:

ObjectGridEventListener

Nested Class Summary

static interface	**ObjectGridEventGroup.ShardEvents** These events are fired when a shard is made a primary shard and when the shard is demoted from a primary.
static interface	**ObjectGridEventGroup.ShardLifecycle** These events are fired when an ObjectGrid shard is initialized and destroyed.
static interface	**ObjectGridEventGroup.TransactionEvents** These events are called every single transaction.

e r f a c e	

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup.ShardLifecycle

Enclosing interface:

ObjectGridEventGroup

`public static interface` **`ObjectGridEventGroup.ShardLifecycle`**

These events are fired when an ObjectGrid shard is initialized and destroyed. A shard can be activated/deactivated multiple times within these two events.

Method Summary

void	**destroy**() Called when the `ObjectGrid` associated with this listener is destroyed.
void	**initialize**(Session session) Invoked when an `ObjectGrid` is initialized.

Method Detail

initialize

`void` **`initialize`**(Session session)

Invoked when an `ObjectGrid` is initialized.

A usable `Session` instance is passed to this listener to provide all of the necessary access to the various ObjectGrid objects.

Parameters:

`session` - a Session instance that this listener is associated with.

See Also:

ObjectGrid.initialize()

---

destroy

`void` **`destroy`**()

Called when the `ObjectGrid` associated with this listener is destroyed.

This method is the opposite of the `initialize` method. When it is called, the listener can free up any resources it uses.

See Also:

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup.ShardEvents

Enclosing interface:

ObjectGridEventGroup

`public static interface` **`ObjectGridEventGroup.ShardEvents`**

These events are fired when a shard is made a primary shard and when the shard is demoted from a primary.

Method Summary

void	**shardActivated**(ObjectGrid grid) This is called when a shard is promoted to a primary.
void	**shardDeactivate**(ObjectGrid grid) This is called when a primary shard is demoted to a replica.

Method Detail

shardActivated

`void` **`shardActivated`**(ObjectGrid grid)

This is called when a shard is promoted to a primary.

Parameters:

`grid` - This is a local reference to the shard containing the primary data.

---

shardDeactivate

`void` **`shardDeactivate`**(ObjectGrid grid)

This is called when a primary shard is demoted to a replica. This can happen is the balancer decides the primary is better placed in a different container. Replication is still active until this method returns to the caller. If any application controlled transactions are in flight then they should be stopped before returning. Once this method returns then any remaining transactions will fail.

Parameters:

`grid` - A reference to the shard.

---

com.ibm.websphere.objectgrid.plugins

Interface ObjectGridEventGroup.TransactionEvents

Enclosing interface:

ObjectGridEventGroup

`public static interface` **`ObjectGridEventGroup.TransactionEvents`**

These events are called every single transaction. These are primarily used when transaction level listening is required. This is usually for pushing changes or invalidation events to peer caches for simple scenarios.

Method Summary

void	**transactionBegin**(String txid, boolean isWriteThroughEnabled) Signals the beginning of a Session transaction.
void	**transactionEnd**(String txid, boolean isWriteThroughEnabled, boolean committed, Collection changes) Signals the ending of a Session transaction.

Method Detail

transactionBegin

```
void transactionBegin(String txid,
                      boolean isWriteThroughEnabled)
```

Signals the beginning of a Session transaction.

A stringified version of the TxID is provided for correlating with the end of the transaction, if so desired. The type of transaction is also provided by the isWriteThroughEnabled boolean parameter.

Parameters:

txid - Stringified version of the TxID

isWriteThroughEnabled - boolean flag indicating whether the Session transaction was started using the Session.beginNoWriteThrough(). method. false is passed if beginNoWriteThrough() was used.

See Also:

Session.begin(), Session.beginNoWriteThrough()

---

transactionEnd

```
void transactionEnd(String txid,
                    boolean isWriteThroughEnabled,
```

```
boolean committed,
Collection changes)
```

Signals the ending of a `Session` transaction.

A string version of the TxID is provided for correlating with the begin of the transaction, if so desired. Map changes are also reported with the collection of `LogSequence`s passed to this method. Typical uses of this event are for customers doing custom peer invalidation or peer commit push. This event listener gives them the changes. Calls to this method are made after commit and are sequenced so that they are delivered one by one, not in parallel. The event order is the commit and rollback order.

Parameters:

`txid` - string version of the TxID
`isWriteThroughEnabled` - boolean flag indicating whether the Session transaction was started using the `Session.beginNoWriteThrough()`. method. `false` is passed if `beginNoWriteThrough()` was used.
`committed` - a boolean flag indicating whether the transaction was committed (`true`) or rolled back (`false`)
`changes` - a Collection of LogSequences representing the changes that were committed or rolled back.

See Also:

`LogSequence.isRollback()`, `Session.begin()`, `Session.beginNoWriteThrough()`, `Session.commit()`, `Session.rollback()`

---

com.ibm.websphere.objectgrid.plugins

Interface MapEventListener

All Superinterfaces:

com.ibm.websphere.objectgrid.plugins.EventListener

```
public interface MapEventListener
extends com.ibm.websphere.objectgrid.plugins.EventListener
```

This callback interface is implemented by the application when it wants to receive events about a Map such as the eviction of a map entry.

Since:

WAS XD 6.0, XC10

See Also:

BackingMap.addMapEventListener(EventListener), BackingMap.removeMapEventListener(EventListener), EventListener

---

Method Summary

void	**entryEvicted**(Object key, Object value) Invoked when the specified entry is evicted from the map.
void	**preloadCompleted**(Throwable t) Invoked when the preloading of this map has completed.

Method Detail

entryEvicted

```
void entryEvicted(Object key,
                  Object value)
```

Invoked when the specified entry is evicted from the map.

The eviction could have occurred either by an `Evictor`'s processing or by invoking one of the `invalidate` methods on the `ObjectMap`.

For a MapEventListener in an `ObjectMap` that is configured to use OutputFormat.RAW for the keys and values, the keys and values objects passed will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the implementation class.

Parameters:

`key` - The key for the map entry that was evicted.
`value` - The value that was in in the map entry evicted. The value object should not be modified.

See Also:

Evictor, EvictionEventCallback, ObjectMap.invalidate(Object, boolean)

---

preloadCompleted

`void` **`preloadCompleted`**(Throwable `t`)

Invoked when the preloading of this map has completed.

This method is useful to determine when a preload operation finishes if asynchronous preloading is enabled. In addition if any error occurred during synchronous or asynchronous preload, it is reported with the invocation of this method.

Parameters:

`t` - A Throwable object that indicates if preload completed without any Throwable occuring during the preload of the map. A null reference indicates preload completed without any Throwable objects occuring during the preload of the map.

See Also:

`Loader.preloadMap(Session, BackingMap)`, BackingMap.setPreloadMode(boolean)

---

com.ibm.websphere.objectgrid.plugins

Interface LogSequenceFilter

public interface **LogSequenceFilter**

This interface can be used to filter a LogSequence. As an operation, such as serialization, needs to know whether a given LogElement should be included or not, this callback object will be used for the boolean check. If the given LogElement should be used in the operation, then "true" should be returned. If the given LogElement should not be used, then "false" should be returned. This interface is primarily used by the `serialize` method of the `LogSequenceTransformer` class.

Since:

WAS XD 6.0, XC10

Method Summary

boolean	**accept**(LogElement logElement) Returns true if the given LogElement should be used; false if the given LogElement should not be used.

Method Detail

accept

boolean **accept**(LogElement logElement)

Returns true if the given LogElement should be used; false if the given LogElement should not be used.

Parameters:

`logElement` - the LogElement to be filtered

Returns:

true if the given LogElement should be used in the operation; false otherwise.

com.ibm.websphere.objectgrid.plugins

Interface LogSequence

All Superinterfaces:

Serializable

```
public interface LogSequence
extends Serializable
```

LogSequence is the ordered list of changes performed against a given map for a given transaction. These changes are recorded as LogElement objects.

Since:

WAS XD 6.0, XC10

Method Summary

Iterator	**getAllChanges**() Returns an iterator for processing all of the changes for a LogSequence.
Iterator	**getChangesByKeys**(Collection keys) Returns an iterator for processing the LogElements that have the requested keys.
Iterator	**getChangesByTypes**(Collection types) Returns an iterator for processing the LogElements that are of the requested type.
String	**getMapName**() Returns the name of the map that these changes apply to.
String	**getObjectGridName**() Returns the name of the ObjectGrid that houses the map that these changes apply to.

Iterator	**getPendingChanges**() Returns an iterator for processing all of the "pending" changes for a LogSequence (for example, pending inserts, updates, and deletes).
boolean	**isDirty**() Returns whether this LogSequence has any LogElements that would "dirty" a Map.
boolean	**isRollback**() Returns whether or not this LogSequence was generated to rollback a transaction.
int	**size**() Returns the total number of LogElements within this LogSequence.

Method Detail

size

`int` **`size`**`()`

Returns the total number of LogElements within this LogSequence.

Returns:
total number of LogElements

---

getPendingChanges

`Iterator` **`getPendingChanges`**`()`

Returns an iterator for processing all of the "pending" changes for a LogSequence (for example, pending inserts, updates, and deletes).

This method is normally used by a `Loader`. A pending change is one that has not been written out to a loader yet using a `flush()` operation. Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Returns:
an Iterator for processing the pending LogElement changes
See Also:
`ObjectMap.flush()`, `Session.flush()`

---

getAllChanges

`Iterator` **`getAllChanges`**`()`

Returns an iterator for processing all of the changes for a LogSequence.

This method would normally be used by an `Evictor` and other plugins that want to know all of the changes introduced by this LogSequence. Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Returns:
an Iterator for processing all of the LogElement changes

---

getChangesByTypes

`Iterator` **`getChangesByTypes`**`(Collection types)`

Returns an iterator for processing the LogElements that are of the requested type. Each member of the input Collection should be one of the defined LogElement Types (INSERT, UPDATE, DELETE, FETCH, TOUCH, or EVICT). Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Parameters:
`types` - A Collection of LogElement Types (INSERT, UPDATE, etc)
Returns:
Iterator for processing all LogElements that support the input Type(s)
Throws:
`IllegalArgumentException` - if types is `null`
See Also:
`LogElement.DELETE`, `LogElement.EVICT`, `LogElement.FETCH`, `LogElement.INSERT`, `LogElement.TOUCH`, `LogElement.UPDATE`, `LogElement.CLEAR`

---

getChangesByKeys

`Iterator` **`getChangesByKeys`**`(Collection keys)`

Returns an iterator for processing the LogElements that have the requested keys.

Note, the returned iterator's `remove()` is not allowed to be called and will throw an exception.

Parameters:
`keys` - a collection of key objects
Returns:
an Iterator for processing all LogElements that match the input key(s)

---

getMapName

`String` **`getMapName`**`()`

Returns the name of the map that these changes apply to.

The caller can use the return value of this method as input to the `Session.getMap(String)` method.

Returns:
The name of the map that these changes apply to
See Also:
`Session.getMap(String)`

---

getObjectGridName

`String` **`getObjectGridName`**`()`

Returns the name of the ObjectGrid that houses the map that these changes apply to.

Returns:
The name of the ObjectGrid that this LogSequence is associated with
Since:
WAS XD 6.0.1

---

isDirty

`boolean` **`isDirty`**`()`

Returns whether this LogSequence has any LogElements that would "dirty" a Map.
That is, if it contains any LogElements of any type other than Fetch/Get, it is considered "dirty".

Returns:
`true` if the LogSequence would modify a Map, if applied; `false` if the LogSequence would not modify a Map, if applied

---

isRollback

`boolean` **`isRollback`**`()`

Returns whether or not this LogSequence was generated to rollback a transaction.

Note, depending on when this LogSequence is used, the transaction itself might already be rolled back.

Returns:
true iff this LogSequence was generated to rollback a transaction.
Since:
WAS XD 6.0.1

---

com.ibm.websphere.objectgrid.plugins

Interface LogElement

All Superinterfaces:

Serializable

```
public interface LogElement
extends Serializable
```

LogElements are the individual entries within a LogSequence. A LogElement has attributes such as operation type (delete, insert, update, etc.), current value, last access time, versioned value, etc. A LogElement is created during a transaction to record in-flight operations. For a LogElement on an `ObjectMap` that is configured to use OutputFormat.RAW for the keys or values, the keys or values objects in the LogElement will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object. To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

Since:

WAS XD 6.0, XC10

See Also:

LogSequence

Nested Class Summary

static class	**LogElement.Type** The `Type` class is used to represent a `LogElement` type.

Field Summary

static LogElement	**CLEAR** The `Type` that represents the CLEAR operation.

[t](#).[T](#)[y](#)[p](#)[e](#)	
static int	**CODE_CLEAR** The type code constant for `CLEAR`.
static int	**CODE_DELETE** The type code constant for `DELETE`.
static int	**CODE_EVICT** The type code constant for `EVICT`.
static int	**CODE_FETCH** The type code constant for `FETCH`.
static int	**CODE_INSERT** The type code constant for INSERT.
static int	**CODE_LOCK** The type code constant for `LOCK`.
static int	**CODE_TOUCH** The type code constant for `TOUCH`.

int	
static int	**CODE_UNDO_NOT_NEEDED** The code constant for UNDO_NOT_NEEDED.
static int	**CODE_UPDATE** The type code constant for UPDATE.
static int	**CODE_UPSERT** The type code constant for UPSERT.
static LogElement.Type	**DELETE** The Type that represents the DELETE operation.
static LogElement.Type	**EVICT** The Type that represents the EVICT operation.

static LogElement.Type	**FETCH** The Type that represents the FETCH operation.
static LogElement.Type	**INSERT** The Type that represents the INSERT operation.
static LogElement.Type	**LOCK** The Type that represents the LOCK operation.
static LogElement.Type	**TOUCH** The Type that represents the TOUCH operation.

[ent.Type](#)	
static [LogElement.Type](#)	**UNDO_NOT_NEEDED** The `Type` that represents an UNDO action is NOT required for this `LogElement`.
static [LogElement.Type](#)	**UPDATE** The `Type` that represents the UPDATE operation.
static [LogElement.Type](#)	**UPSERT** The `Type` that represents the UPSERT operation.

Method Summary

Object	**getAfterImage**() Gets the "after image" value object.
Object	**getBeforeImage**() Gets the "before image" of the value object.
CacheEntry	**getCacheEntry**() Returns the `CacheEntry` for this key.
Object	**getCurrentValue**() Gets the value for this `LogElement`.
Object	**getKey**() Returns the key for this LogElement.
long	**getLastAccessTime**() Returns the last access time associated with this `LogElement`.
LogElement.Type	**getType**() Gets the type of this `LogElement`.
LogElement.Type	**getUndoType**() Returns what operation must be performed to "undo" a prior change the transaction made to the map entry.

[e](#)	
[Object](#)	**getVersionedValue**() Gets the versioned object at the time the object was first associated with the transaction.
boolean	**isCascaded**() Answers `true` if this LogElement is a result of a cascade operation.
boolean	**isPending**() Answers `true` if this change has NOT been applied to the loader.
void	**setVersionedValue**([Object](#) v) Used to update the versioned object after an update of map entry occurs.

Field Detail

CODE_INSERT

```
static final int CODE_INSERT
```

The type code constant for INSERT.

See Also:

[INSERT](#), [LogElement.Type.getCode()](#), Constant Field Values

---

CODE_UPDATE

```
static final int CODE_UPDATE
```

The type code constant for `UPDATE`.

See Also:

[UPDATE](#), [LogElement.Type.getCode()](#), Constant Field Values

---

CODE_DELETE

```
static final int CODE_DELETE
```

The type code constant for `DELETE`.

See Also:

[DELETE](#), [LogElement.Type.getCode()](#), Constant Field Values

---

CODE_EVICT

`static final int` **`CODE_EVICT`**

The type code constant for `EVICT`.

See Also:
`EVICT`, `LogElement.Type.getCode()`, Constant Field Values

CODE_FETCH

`static final int` **`CODE_FETCH`**

The type code constant for `FETCH`.

See Also:
`FETCH`, `LogElement.Type.getCode()`, Constant Field Values

CODE_TOUCH

`static final int` **`CODE_TOUCH`**

The type code constant for `TOUCH`.

See Also:
`TOUCH`, `LogElement.Type.getCode()`, Constant Field Values

CODE_CLEAR

`static final int` **`CODE_CLEAR`**

The type code constant for `CLEAR`.

Since:
WAS XD 6.1.0.3
See Also:
`CLEAR`, `LogElement.Type.getCode()`, Constant Field Values

CODE_LOCK

`static final int` **`CODE_LOCK`**

The type code constant for `LOCK`.

Since:
8.6, XC10 2.5
See Also:
`LOCK`, `LogElement.Type.getCode()`, Constant Field Values

CODE_UPSERT

`static final int` **`CODE_UPSERT`**

The type code constant for `UPSERT`.

Since:
8.6, XC10 2.5
See Also:
`UPSERT`, `LogElement.Type.getCode()`, Constant Field Values

CODE_UNDO_NOT_NEEDED

`static final int` **`CODE_UNDO_NOT_NEEDED`**

The code constant for `UNDO_NOT_NEEDED`. Used to indicate no operation is needed to undo the changes for this `LogElement` since this `LogElement` was never processed or it was an operation that does not require an undo operation.

See Also:
Constant Field Values

INSERT

`static final LogElement.Type` **`INSERT`**

The `Type` that represents the INSERT operation.

UPDATE

`static final LogElement.Type` **`UPDATE`**

The `Type` that represents the UPDATE operation.

DELETE

`static final LogElement.Type` **`DELETE`**

The `Type` that represents the DELETE operation.

EVICT

`static final LogElement.Type` **`EVICT`**

The `Type` that represents the EVICT operation.

FETCH

`static final LogElement.Type` **`FETCH`**

The `Type` that represents the FETCH operation.

TOUCH

`static final LogElement.Type` **`TOUCH`**

The `Type` that represents the TOUCH operation.

CLEAR

`static final LogElement.Type` **`CLEAR`**

The `Type` that represents the CLEAR operation.

Since:
WAS XD 6.1.0.3

LOCK

static final LogElement.Type **LOCK**

The Type that represents the LOCK operation.

Since:
8.6, XC10 2.5

UPSERT

static final LogElement.Type **UPSERT**

The Type that represents the UPSERT operation.

Since:
8.6, XC10 2.5

UNDO_NOT_NEEDED

static final LogElement.Type **UNDO_NOT_NEEDED**

The Type that represents an UNDO action is NOT required for this LogElement.

Method Detail

getType

LogElement.Type **getType**()

Gets the type of this LogElement. The type indicates what operation needs to be applied to the map entry.

Returns:
the type of this LogElement.

---

getCurrentValue

Object **getCurrentValue**()

Gets the value for this LogElement.

For a LogElement on an ObjectMap that is configured to use a ValueSerializerPlugin, the values in the LogSequence will be SerializedValue objects. If required, you can use the SerializedEntry.getObject() method to retrieve (possibly inflating the serialized object) the original value object.

The original value represents the new value that should be applied to the BackingMap and Loader. This value can be cast to ValueProxyInfo when a value interface is in use in order to determine the set of dirty attributes.

Returns:
the value in case of INSERT, UPDATE, UPSERT, or FETCH, null in the case of DELETE or EVICT.

See Also:
ValueProxyInfo

---

getCacheEntry

`CacheEntry` **`getCacheEntry`**`()`

Returns the `CacheEntry` for this key. The key, current committed value, etc. can be accessed from the `CacheEntry`.

Returns:
the entry in the cache that is requested to be updated.
See Also:
`CacheEntry.getCommittedValue()`, `getKey()`

---

isPending

`boolean` **`isPending`**`()`

Answers `true` if this change has NOT been applied to the loader.

Changes can previously be applied to a loader using the `ObjectMap.flush()` or `Session.flush()` methods. This method reveals whether the change in this `LogElement` has already been applied to the Loader using one of those methods.

Returns:
`true` if this change has NOT been applied to the loader.
See Also:
`ObjectMap.flush()`, `Session.flush()`

---

getVersionedValue

`Object` **`getVersionedValue`**`()`

Gets the versioned object at the time the object was first associated with the transaction.

For a LogElement on an `ObjectMap` that is configured to use a or `ValueSerializerPlugin`, the versioned object will be returned as an `XsDataInputStream`, read will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object. For a LogElement on an `ObjectMap` that is configured to use a `ValueSerializerPlugin` that generates version objects, the version object will be the data stream representing the data.

Returns:
The versioned object.
See Also:
`OptimisticCallback`

---

setVersionedValue

`void` **`setVersionedValue`**`(``Object`` v)`

Used to update the versioned object after an update of map entry occurs.

The Loader can use this method when it is using an optimistic strategy and uses the `OptimisticCallback.updateVersionedObjectForValue(Object)` method to get an updated version object.

Parameters:
`v` - The versioned object.
See Also:
`OptimisticCallback.updateVersionedObjectForValue(Object)`

getLastAccessTime

`long` **`getLastAccessTime`**`()`

Returns the last access time associated with this `LogElement`.

Returns:
last access time

---

getUndoType

`LogElement.Type` **`getUndoType`**`()`

Returns what operation must be performed to "undo" a prior change the transaction made to the map entry.

Note, an undo type of `UNDO_NOT_NEEDED` is returned if nothing needs to be undone for this `LogElement`.

Returns:
the "undo" type of this `LogElement`. It can be one of: INSERT, UPDATE, DELETE or UNDO_NOT_NEEDED

---

getBeforeImage

`Object` **`getBeforeImage`**`()`

Gets the "before image" of the value object.

The "before image" is the value object that existed in map entry prior to applying a change to map entry. Note, it is possible for a `null` reference to be returned (e.g. in the case where a new map entry is created).

For a LogElement on an `ObjectMap` that is configured to use `OutputFormat.RAW` for the values, the value will be a `SerializedValue` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

Returns:
the value prior to applying the change

---

getAfterImage

`Object` **`getAfterImage`**`()`

Gets the "after image" value object.

The "after image" is the value object that existed in map entry after applying a change to the map entry. Note, it is possible for a `null` reference to be returned (e.g. in the case where an existing map entry is removed/evicted).

For a LogElement on an `ObjectMap` that is configured to use `OutputFormat.RAW` for the value, the value will be a `SerializedValue` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original value object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

Returns:
the value after applying the change

---

isCascaded

`boolean` **`isCascaded`**`()`

Answers `true` if this LogElement is a result of a cascade operation. This only applies to ObjectGrid EntityManager programming model.

ObjectGrid EntityManager supports cascade operations. For example, when persisting an entity P, if P has a relation to entity C with `CascadeType.PERSIST` enabled, C will also be persisted as a result of the cascade operation. The method isCascaded() returns true for the LogElement object which represents C, and the method returns false for the LogElement object which represents P.

Returns:
`true` if the LogElement object is a result of cascade operation.
Since:
6.1.0.5 FIX1
See Also:
`EntityManager`

---

getKey

`Object` **`getKey`**`()`

Returns the key for this LogElement.

For a LogElement on an `ObjectMap` that is configured to use `OutputFormat.RAW` for the keys, the value will be a `SerializedKey` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key object.

To override the map's output format configuration, use the `PluginOutputFormat` annotation in the caller of the LogElement.

This method can be used instead of LogElement.getCacheEntry().getKey().

Returns:
the key for this LogElement.
Since:
7.0
See Also:
`CacheEntry.getKey()`

---

com.ibm.websphere.objectgrid.plugins

Class LogElement.Type

java.lang.Object
  com.ibm.websphere.objectgrid.plugins.LogElement.Type

All Implemented Interfaces:
Comparable

Enclosing interface:
LogElement

```
public static class LogElement.Type
extends Object
implements Comparable
```

The `Type` class is used to represent a `LogElement` type.

Since:
WAS XD 6.0

Method Summary

int	**compareTo**(Object object)
int	**getCode**() Gets the type code for this object.
String	**toString**()
static LogElement.Type	**valueOf**(int typeCode) Return the Type for a given type code.

p e	

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Method Detail

getCode

```
public int getCode()
```

Gets the type code for this object.

Returns:
the type code

---

compareTo

```
public int compareTo(Object object)
```

Specified by:
compareTo in interface Comparable

See Also:
Comparable.compareTo(Object)

---

toString

```
public String toString()
```

Overrides:
toString in class Object

See Also:
Object.toString()

---

valueOf

```
public static LogElement.Type valueOf(int typeCode)
```

Return the Type for a given type code.

Parameters:
typeCode - the typecode
Returns:
the Type
Throws:
IllegalArgumentException - if the typeCode isn't valid.
Since:
8.6, XC10 2.5

---

com.ibm.websphere.objectgrid.plugins

Class LoaderException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

UnavailableServiceException

---

public class **LoaderException**
extends ObjectGridException

This exception is the base exception for any exceptions encountered by a `Loader`.

Since:

WAS XD 6.0, XC10

See Also:

`Loader`, Serialized Form

Constructor Summary

LoaderException() Constructs a new LoaderException with `null` as its detail message.
LoaderException(String message) Constructs a new LoaderException with the specified detail message.
LoaderException(String message, Throwable cause) Constructs a new LoaderException with the specified detail message and cause.
LoaderException(Throwable cause) Constructs a new LoaderException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LoaderException

public **LoaderException**()

Constructs a new LoaderException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

LoaderException

public **LoaderException**(String message)

Constructs a new LoaderException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
ObjectGridException.initCause(Throwable), Throwable.getMessage()

LoaderException

public **LoaderException**(String message,
Throwable cause)

Constructs a new LoaderException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LoaderException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
ObjectGridException.getCause(), Throwable.getMessage()

LoaderException

public **LoaderException**(Throwable cause)

Constructs a new LoaderException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LoaderExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for

later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid.plugins

Interface EvictorData

All Known Implementing Classes:

EvictorData.SpecialEvictorData

`public interface` **`EvictorData`**

This interface is optionally used by an implementator of the `Evictor` interface. Application changes applied to a `BackingMap` are asynchronous from the `Evictor` activity. The `Evictor` is not notified of changes to the `BackingMap` until after application transactions are committed. Consequently, if an `Evictor` decides to evict a map entry, it is possible that the `BackingMap` could evict an entry that was different from the original entry it was tracking. For example, consider that an application could execute a transaction that removes a map entry. Before the `Evictor` is notified of the remove, another transaction inserts a new entry into the `BackingMap` for the same key as the old entry. Consequently, the `Evictor` could evict the newly created entry when it meant to evict the old entry. To help close this small timing window, the `Evictor` can use this interface to associate evictor specific data with a map entry. The `Evictor` can then do the following:

- store the `EvictorData` object for a map entry by using the `EvictionEventCallback.setEvictorData(Object, EvictorData)` method.
- retrieve the `EvictorData` object for a map entry by using the `EvictionEventCallback.getEvictorData(Object)` method.
- Conditionally evict a map entry if and only if the cache entry for a specified key has the exact same `EvictorData` object (the java == operator returns true) associated with it by using the `EvictionEventCallback.evictMapEntries(List)` method.

Since:
WAS XD 6.0.1, XC10

See Also:
`Evictor`, `EvictionEventCallback`

Nested Class Summary

static class	**EvictorData.SpecialEvictorData** Special value class used for representing the key not being found in the `BackingMap`.

Field Summary

static	

static [EvictorData](#)	**KEY_NOT_FOUND** A special value indicating that the key was not found.

Method Summary

[Object](#)	**getKey**() Retrieves the key object for this `EvictorData` instance.

Field Detail

KEY_NOT_FOUND

`static final EvictorData` **KEY_NOT_FOUND**

A special value indicating that the key was not found.

Method Detail

getKey

`Object` **getKey**()

Retrieves the key object for this `EvictorData` instance.

Returns:

the same key object that was passed to the `EvictionEventCallback.setEvictorData(Object, EvictorData)` method when this EvictorData was associated with the map entry with the given key.

See Also:

`EvictionEventCallback.setEvictorData(Object, EvictorData)`

com.ibm.websphere.objectgrid.plugins

Class EvictorData.SpecialEvictorData

java.lang.Object

  com.ibm.websphere.objectgrid.plugins.EvictorData.SpecialEvictorData

All Implemented Interfaces:

EvictorData

Enclosing interface:

EvictorData

```
public static final class EvictorData.SpecialEvictorData
extends Object
implements EvictorData
```

Special value class used for representing the key not being found in the `BackingMap`.

Since:

WAS XD 6.0.1

Nested Class Summary

Nested classes/interfaces inherited from interface com.ibm.websphere.objectgrid.plugins.EvictorData
EvictorData.SpecialEvictorData

Field Summary

Fields inherited from interface com.ibm.websphere.objectgrid.plugins.EvictorData
KEY_NOT_FOUND

Constructor Summary

EvictorData.SpecialEvictorData()

Method Summary

Object	**getKey**() Dummy implementation method since this class will not be called.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EvictorData.SpecialEvictorData

public **EvictorData.SpecialEvictorData**()

Method Detail

getKey

public Object **getKey**()

Dummy implementation method since this class will not be called.

Specified by:
getKey in interface EvictorData

Returns:
null

See Also:
EvictionEventCallback.setEvictorData(Object, EvictorData)

---

com.ibm.websphere.objectgrid.plugins

Interface Evictor

All Known Implementing Classes:

LFUEvictor, LRUEvictor

`public interface` **`Evictor`**

Data contained in a `BackingMap` are evicted when the map is full. This plugin is used by the `BackingMap` to determine when and what to evict from the map based on some algorithm (LRU, LFU, time based, etc).

An `Evictor` implementation that also implements the `BackingMapLifecycleListener` interface will be automatically added as an `EventListener` on the `BackingMap` when the evictor set on the backing map.

An `Evictor` may also implement the `BackingMapPlugin` interface in order to receive enhanced BackingMap plug-in lifecycle method calls. The plug-in is then also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Since:

WAS XD 6.0, XC10

See Also:

BackingMap.addMapEventListener(EventListener), BackingMap.setEvictor(Evictor), EvictorData

---

Method Summary

void	**activate**() This method is called to activate the Evictor.
void	**apply**(LogSequence sequence) Called after a transaction has committed to allow the evictor to track object usage in the `BackingMap`.
void	**deactivate**() This method is called to deactivate the Evictor.
void	**destroy**() Called when the `BackingMap` associated with this evictor is destroyed.
void	**initialize**(BackingMap map, EvictionEventCallback callback) Called by a `BackingMap` instance during the evictor initialization time.

Method Detail

initialize

```
void initialize(BackingMap map,
                EvictionEventCallback callback)
```

Called by a `BackingMap` instance during the evictor initialization time.

The `BackingMap` calls this method so the Evictor instance can have references to the `BackingMap` and `EvictionEventCallback` instances. The evictor can signal events to have specific entries evicted using the `EvictionEventCallback`.

Parameters:

`map` - the BackingMap instance
`callback` - the EvictionEventCallback instance

See Also:

BackingMap, EvictionEventCallback

---

destroy

```
void destroy()
```

Called when the `BackingMap` associated with this evictor is destroyed.

This method is the opposite of the `initialize` method. When it is called, the Evictor can free up any resources it uses.

See Also:

ObjectGrid.destroy()

---

apply

```
void apply(LogSequence sequence)
```

Called after a transaction has committed to allow the evictor to track object usage in the `BackingMap`.

This method also reports any entries that have been successfully evicted. Note, this method is not called for transactions that are rolled back. If there is a need to track object usage for rolled back transactions, the evictor must implement the `RollbackEvictor` interface as well.

This method is called after a transaction has completed. Consequently, all transaction locks that were acquired by the completed transaction are no longer held. Potentially, multiple threads could call this method concurrently and each thread would be completing its own transaction. Since transaction locks are already released by the completed transaction, this method must provide its own synchronization to ensure it is thread safe. For an Evictor in an `ObjectMap` that is configured to use `OutputFormat.RAW` for the keys or values, the keys and values objects in the LogSequence will be `SerializedKey` or `SerializedValue` objects respectively. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key or value object. To override the map's output format configuration, use the `PluginOutputFormat` annotation in the implementation class.

Parameters:

`sequence` - the `LogSequence` of changes committed to the map

See Also:

`RollbackEvictor`

---

activate

`void **activate**()`

This method is called to activate the Evictor. Until this method is called, the Evictor must not use the EvictionEventCallback interface to evict any map entries. If it does use the EvictionEventcallback interface to evict map entries prior to activate being called, an IllegalStateException is thrown.

---

deactivate

`void **deactivate**()`

This method is called to deactivate the Evictor. Once this method is called, the Evictor must quit using the EvictionEventCallback interface to evict any map entries. If it does use the EvictionEventcallback interface after this method is called, an IllegalStateException is thrown.

---

com.ibm.websphere.objectgrid.plugins

Interface EvictionEventCallback

public interface **EvictionEventCallback**

An instance of `EvictionEventCallback` is passed into the `Evictor` at initialization time. When an eviction method is called, corresponding methods of `EvictionEventCallback` will be called so the `BackingMap` can process evictions.

Since:

WAS XD 6.0, XC10

See Also:

`Evictor`, `EvictorData`

Method Summary

void	**evictEntries**(List keysToEvictList) If an `Evictor` chooses not to implement the `EvictorData` interface, this method can be used to evict a map entry.
void	**evictMapEntries**(List evictorDataList) This method is the preferred method for the `Evictor` to use when evicting map entries.
EvictorData	**getEvictorData**(Object key) Gets the evictor data for a specified BackingMap cache entry.
void	**setEvictorData**(Object key, EvictorData data) Sets the evictor data for a specified BackingMap cache key.

Method Detail

setEvictorData

void **setEvictorData**(Object key,
EvictorData data)

Sets the evictor data for a specified BackingMap cache key.

This method can be used by an implementor of the `Evictor` interface to keep data that the evictor needs for determining which cache entry to evict.

Parameters:

`key` - is the key for accessing a BackingMap entry.

`data` - the EvictorData object to store as evictor data for a specified key.

Throws:

`IllegalArgumentException` - if key is a null reference or there is no BackingMap cache entry for this key.

Since:

WAS XD 6.0.1

See Also:

`Evictor`

---

getEvictorData

`EvictorData` **`getEvictorData`**`(Object key)`

Gets the evictor data for a specified BackingMap cache entry.

Parameters:

`key` - the key for the BackingMap entry to set.

Returns:

if the specified key is not found in BackingMap, then the special value `EvictorData.KEY_NOT_FOUND` is returned. If the key is found in the BackingMap, the same reference that was previously passed to the `setEvictorData(Object, EvictorData)` method of this interface is returned. A null reference is returned if the key is found, but the `setEvictorData` method was not previously called for the specified key.

Throws:

`IllegalArgumentException` - if key is a null reference.

Since:

WAS XD 6.0.1

See Also:

`setEvictorData(Object, EvictorData)`, `EvictorData.KEY_NOT_FOUND`

---

evictMapEntries

```
void evictMapEntries(List evictorDataList)
                     throws ObjectGridException
```

This method is the preferred method for the `Evictor` to use when evicting map entries. A list of `EvictorData` objects is passed as an argument to this method. For each `EvictorData` object in the list, the key is obtained from the `EvictorData` object and used to determine which BackingMap entry to evict. The BackingMap entry is evicted if and only if the cache entry for BackingMap entry contains the exact same `EvictorData` object in it. That is, the java == operator is used to ensure it is the exact same `EvictorData` object. If the == operator indicates a different object, then the map entry is not evicted. For those map entries that are physically evicted from the map, the `Evictor` will receive notification through its `apply` method.

Parameters:

`evictorDataList` - a list of `EvictorData` objects to process. The caller must guarantee this parameter is not `null` or contain any `null` references.

Throws:

`ObjectGridException` - if an error occurs during processing

`ClassCastException` - if an object in evictorDataList does not implement the `EvictorData` interface.

Since:

WAS XD 6.0.1

See Also:

Evictor.apply(LogSequence), EvictorData.getKey()

---

evictEntries

```
void evictEntries(List keysToEvictList)
                  throws ObjectGridException
```

If an `Evictor` chooses not to implement the `EvictorData` interface, this method can be used to evict a map entry. However, the `Evictor` must be prepared to handle the exposure of an application removing and recreating a map entry before the `Evictor` has an opportunity to call this method.

For this method, a list of map keys is passed. The list is evaluated and an eviction is conducted on the list. When the entries are physically evicted from the map, the `Evictor` will receive notification through its `apply` method.

Parameters:

`keysToEvictList` - List of keys to evict from the map. The caller must guarantee this parameter is not `null` or contain any `null` references.

Throws:

`ObjectGridException` - if an error occurs during processing

See Also:

`Evictor.apply(LogSequence)`

---

com.ibm.websphere.objectgrid.plugins

Class CacheEntryException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.CacheEntryException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **CacheEntryException**
extends ObjectGridException

This exception indicates an error occurred during a cache entry operation.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

CacheEntryException() Constructs a new CacheEntryException with `null` as its detail message.
CacheEntryException(String message) Constructs a new CacheEntryException with the specified detail message.
CacheEntryException(String message, Throwable cause) Constructs a new CacheEntryException with the specified detail message and cause.
CacheEntryException(Throwable cause) Constructs a new CacheEntryException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CacheEntryException

`public` **`CacheEntryException`**`()`

Constructs a new CacheEntryException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

CacheEntryException

`public` **`CacheEntryException`**`(String message)`

Constructs a new CacheEntryException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

CacheEntryException

`public` **`CacheEntryException`**`(String message,`
`Throwable cause)`

Constructs a new CacheEntryException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this CacheEntryException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

CacheEntryException

`public` **`CacheEntryException`**`(Throwable cause)`

Constructs a new CacheEntryException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for CacheEntryExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.
See Also:
`ObjectGridException.getCause()`

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

com.ibm.websphere.objectgrid.plugins

Interface CacheEntry

All Superinterfaces:

Serializable

```
public interface CacheEntry
extends Serializable
```

This interface represents a cache entry in an ObjectGrid map.

Since:

WAS XD 6.0, XC10

Method Summary

Object	**getCommittedValue**() Returns the committed value for this entry.
Object	**getKey**() Returns the key for this entry.
long	**getLastAccessTime**() Returns the last time this entry was accessed.
long	**getTTL**() Returns the time-to-live value for this entry.
boolean	**isInBackingMap**() Indicates whether this entry is in the `BackingMap`

Method Detail

isInBackingMap

```
boolean isInBackingMap()
```

Indicates whether this entry is in the `BackingMap`

Returns:
Returns `true` if this element is in the BackingMap

---

getKey

`Object` **`getKey`**`()`

Returns the key for this entry.

For a CacheEntry on an `ObjectMap` that is configured to use a `KeySerializerPlugin`, the value will be a `SerializedKey` object. If required, you can use the `SerializedEntry.getObject()` method to retrieve (possibly inflating the serialized object) the original key object.

Returns:
the key

---

getCommittedValue

`Object` **`getCommittedValue`**`()`

Returns the committed value for this entry.

The type of the object returned from the `getCommittedValue()` method depends on various configuration and storage options used by the `ObjectMap` that holds the CacheEntry. In the default case, `getCommittedValue()` returns the Java object of the same type that was put into the map. For an `ObjectMap` that is configured to use a `ValueSerializerPlugin`, the committed value depends on the underlying storage mechanism, typically represented as an array of bytes.

Returns:
the committed value

---

getTTL

`long` **`getTTL`**`()`

Returns the time-to-live value for this entry.

Returns:
the time-to-live value

---

getLastAccessTime

`long` **`getLastAccessTime`**`()`

Returns the last time this entry was accessed.

Returns:
last access time.

---

com.ibm.websphere.objectgrid.plugins

Interface BeanFactory

public interface **BeanFactory**

Implement this interface to allow bean factories like Spring or Google guice to be integrated. This allows ObjectGrid to delegate to an external Bean Factory to instantiate beans needed by ObjectGrid.

Since:

WAS XD 6.1 FIX3, XC10

Method Summary

Object	**getBean**(String name) This returns an instance of the bean with the specified name.

Method Detail

getBean

Object **getBean**(String name)

This returns an instance of the bean with the specified name.

Parameters:

name - The name of the bean instance to return.

Returns:

the bean instance.

Package com.ibm.websphere.objectgrid.security

This package has the class MapPermission and class AdminPermission which represents the permissions for to access the ObjectGrid maps and ObjectGrid administration respectively.

See:

Description

Class Summary	
SecurityConstants	This class contains the constants used for security configuration.

Exception Summary	
ObjectGridSecurityException	This exception represents a general ObjectGrid security exception.

Package com.ibm.websphere.objectgrid.security Description

This package has the class MapPermission and class AdminPermission which represents the permissions for to access the ObjectGrid maps and ObjectGrid administration respectively.

MapPermission action types.

The ObjectGrid defines 5 permission actions that are used to authorize accesses to the maps. These permissions allow access to maps to be controlled by an administrator. Objects within the ObjectGrid use a simple naming scheme. Each Map is named using the convention of the ObjectGrid name followed by a period followed by the Map name. For example, if the object grid name is "myObjectGrid" and the map name is "myMap", then the map name used in the permission is "myobjectgrid.mymap".

Wildcards can be used on names with some restrictions. A wild card "*" can be used to replace the map name or the object grid name, but not partially. For example, "myObjectGrid.*", "*.myMap", and "*.*" are valid names, but "myObject*.*" is not valid.

There are five actions with the permission object ObjectMapPermission.

- Read
  This action allows get operations to be issued against a Map.
- Write
  This action allows put operations to be issued against a Map. It allows existing entries to be updated.
- Remove
  This action allows entries to be removed from the Map.
- Insert
  This action allows clients to add entries to a Map.

- Invalidate
  This action allows clients to invalidate entries from the Map.

	com.ibm.websphere.objectgrid.ObjectMap/ com.ibm.websphere.objectgrid.JavaMap
Read	boolean containsKey(Object)
	boolean equals(Object)
	Object get(Object)
	Object get(Object, Serializable)
	List getAll(List)
	List getAll(List keyList, Serializable)
	List getAllForUpdate(List)
	List getAllForUpdate(List, Serializable)
	Object getForUpdate(Object)
	Object getForUpdate(Object, Serializable)
write	Object put(Object key, Object value)
	void put(Object, Object, Serializable)
	void putAll(Map)
	void putAll(Map, Serializable)
	void update(Object, Object)
	void update(Object, Object, Serializable)
insert	public void insert(Object, Object)
	void insert(Object, Object, Serializable)
	remove Object remove(Object)
	void removeAll(Collection)
invalidate	public void invalidate(Object, boolean)
	void invalidateAll(Collection, boolean)
	int setTimeToLive(int)

An authroizationMechanism setting of the ObjectGrid has two possible values: JAAS and custom. Users can also use API `ObjectGrid.setAuthorizationMechanism(int)` to set which authorization mechanism the object grid will use.

A value "JAAS" means ObjectGrid will rely on JAAS authorization mechanism to handle the authorization. A JAAS policy file should be configured to associate permissions with a set of credentials and/or groups of credentials. We recommend that groups should be used as then new users can be added to groups without modifying the policy file.

A value "custom" means ObjectGrid will rely on custom authorization mechanism to handle the authorization. Users can set call `ObjectGrid.setObjectGridAuthorization(com.ibm.websphere.objectgrid.security.plugins.ObjectGridAuthorization ogAuthorization)` to set their custom authorization plug-in. Users can also configure the objectgrid.xml to achieve the same result.

AdminPermission types

An AdminPermission has two types: ADMIN and MONITOR. An AdminPermission with ADMIN name grants permissions to access all the ManagementMBean methods. An AdminPermission with MONITOR name grants permissions to access the ManagementMBean read-only methods. Therefore, ADMIN permission implies MONITOR permission.

The detailed operations granted to users with different permissions are listed in the following

table. These operations correspond to the methods in the ManagementMBean interface:

operations	admin	monitor
startServer	Y	N
stopServer	Y	N
forceStopServer	Y	N
setServerTrace	Y	N
retrieveServerStatus	Y	Y
getMapStats	Y	Y
getOGStats	Y	Y
getReplicationStats	Y	Y

The table can read like this: If the client has admin permission, it can execute "startServer" task; if the client has monitor permission, it cannot execute "startServer" task.

AgentPermission types

An AgentPermission represents permissions to the datagrid agents. The name of the permission is the full name of the ObjectGrid map, and the action is a "," delimited string of agent implementation class names or package names.

The following methods in the class `AgentManager` requires AgentPermission:

- `AgentManager.callMapAgent(MapGridAgent, Collection)`
- `AgentManager.callMapAgent(MapGridAgent)`
- `AgentManager.callReduceAgent(ReduceGridAgent, Collection)`
- `AgentManager.callReduceAgent(ReduceGridAgent, Collection)`

ObjectGridPermission types

An ObjectGridPermission represents permissions to an ObjectGrid. The name of the permission is the ObjectGrid name, and the action is either "query" or "dynamicmap".

The detailed methods which require different permissions are listed in the following table:

methods	action
`Session.createObjectQuery(String)`	query
`EntityManager.createQuery(String)`	query
`Session.getMap(String)`	dynamicmap

ServerMapPermission types

An ServerMapPermission represents permissions to an ObjectMap hosted in a server. The name of the permission is the full name of the ObjectGrid map name, and the action is either "replicate" or "dynamicIndex".

The detailed methods which require different ServerMapPermission are listed in the following table:

methods	action
`ClientReplicableMap.enableClientReplication(Mode, int[], ReplicationMapListener)`	replicate
`BackingMap.createDynamicIndex(String, boolean, String, DynamicIndexCallback)`	dynamicIndex
`BackingMap.removeDynamicIndex(String)`	dynamicIndex

SecurityConstants

SecurityConstants class contains constants used for representing the security parameters.

---

com.ibm.websphere.objectgrid.security

Class SecurityConstants

java.lang.Object

↳ **com.ibm.websphere.objectgrid.security.SecurityConstants**

---

public class **SecurityConstants**
extends Object

This class contains the constants used for security configuration.

Since:

WAS XD 6.0, XC10

Field Summary

static int	**ACCESS_BY_CREATOR_ONLY_COMPLEMENT** The access by creator only authorization is enabled to complement the ObjectGrid map authorization.
static int	**ACCESS_BY_CREATOR_ONLY_DISABLED** The access by creator only authorization is disabled.
static int	**ACCESS_BY_CREATOR_ONLY_SUPERSEDE** The access by creator only authorization is enabled to supersede the ObjectGrid map authorization.
static int	**AUTHORIZATION_MECHANISM_CUSTOM** Constant representing custom authorization
st	

static int	**AUTHORIZATION_MECHANISM_JAAS** Constant representing JAAS authorization
static int	**CLIENT_CERTIFICATE_AUTHENTICATION_NEVER** Constant indicating client certificate authentication is not supported.
static int	**CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED** Constant indicating client certificate authentication is required.
static int	**CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED** Constant indicating client certificate authentication is supported.
static int	**CREDENTIAL_AUTHENTICATION_NEVER** Constant indicating credential authentication is not supported.
static int	**CREDENTIAL_AUTHENTICATION_REQUIRED** Constant indicating credential authentication is required.
static int	**CREDENTIAL_AUTHENTICATION_SUPPORTED** Constant indicating credential authentication is supported.
static	**NEVER_STRING**

static String	String representation for value Never indicating an option is not supported.
static String	**NEW_SECURE_TOKEN_MANAGER_STRING** String representation for "autoSecret" type of the secure token manager.
static String	**REQUIRED_STRING** String representation for value Required indicating an option is required.
static String	**SECURE_TOKEN_MANAGER_CUSTOM_STRING** String representation for "custom" type of the secure token manager.
static String	**SECURE_TOKEN_MANAGER_DEFAULT_STRING** String representation for "default" type of the secure token manager.
static String	**SECURE_TOKEN_MANAGER_NONE_STRING** String representation for "none" type of the secure token manager.
static int	**SSL_REQUIRED** Constant indicating SSL transport is required.

int	
static String	**SSL_REQUIRED_STRING** String representation for value SSL-Required indicating SSL transport type is required.
static int	**SSL_SUPPORTED** Constant indicating SSL transport is supported.
static String	**SSL_SUPPORTED_STRING** String representation for value SSL-Supported indicating SSL transport type is supported.
static String	**SUPPORTED_STRING** String representation for value Supported indicating an option is supported.
static int	**TCP_IP** Constant indicating TCP/IP is the only supported transport.
static String	**TCPIP_STRING** String representation for value TCP/IP indicating a transport type of TCP/IP is used.

Constructor Summary

SecurityConstants()

Method Summary

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

AUTHORIZATION_MECHANISM_JAAS

public static final int **AUTHORIZATION_MECHANISM_JAAS**

Constant representing JAAS authorization

See Also:
ObjectGrid.setAuthorizationMechanism(int), Constant Field Values

---

AUTHORIZATION_MECHANISM_CUSTOM

public static final int **AUTHORIZATION_MECHANISM_CUSTOM**

Constant representing custom authorization

See Also:
ObjectGrid.setAuthorizationMechanism(int), Constant Field Values

---

TCP_IP

public static final int **TCP_IP**

Constant indicating TCP/IP is the only supported transport.

If the client's transport type is set to this value, TCP/IP is the only supported transport type. If the server requires SSL, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1
See Also:
ClientSecurityConfiguration.setTransportType(int), Constant Field Values

---

SSL_SUPPORTED

public static final int **SSL_SUPPORTED**

Constant indicating SSL transport is supported.

If the client's transport type is set to this value, the client supports both TCP/IP and SSL. SSL will be used if both sides side supports SSL. Otherwise, TCP/IP will be used.

Since:
WAS XD 6.0.1
See Also:
ClientSecurityConfiguration.setTransportType(int), Constant Field Values

SSL_REQUIRED

`public static final int` **`SSL_REQUIRED`**

Constant indicating SSL transport is required.

If the client's transport type is set to this value, SSL is the only supported transport type. If the server requires TCP/IP, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setTransportType(int)`, Constant Field Values

CREDENTIAL_AUTHENTICATION_NEVER

`public static final int` **`CREDENTIAL_AUTHENTICATION_NEVER`**

Constant indicating credential authentication is not supported.

If the credential authentication type is set to this value, no credential authentication will be enforced. If the server requires credential authentication, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setCredentialAuthenticationType(int)`, Constant Field Values

CREDENTIAL_AUTHENTICATION_SUPPORTED

`public static final int` **`CREDENTIAL_AUTHENTICATION_SUPPORTED`**

Constant indicating credential authentication is supported.

If the credential authentication type is set to this value, credential authentication will be enforced if and only if both client and server support credential authentication.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setCredentialAuthenticationType(int)`, Constant Field Values

CREDENTIAL_AUTHENTICATION_REQUIRED

`public static final int` **`CREDENTIAL_AUTHENTICATION_REQUIRED`**

Constant indicating credential authentication is required.

If the credential authentication type is set to this value, credential authentication will be enforced. If the server doesn't support credential authentication, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setCredentialAuthenticationType(int)`, Constant Field Values

CLIENT_CERTIFICATE_AUTHENTICATION_NEVER

`public static final int` **`CLIENT_CERTIFICATE_AUTHENTICATION_NEVER`**

Constant indicating client certificate authentication is not supported.

If the client certificate authentication type is set to this value, no client certificate authentication will be enforced. If the server doesn't support client certificate authentication, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setClientCertificateAuthentication(int)`, Constant Field Values

CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED

`public static final int` **`CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED`**

Constant indicating client certificate authentication is supported.

If the client certificate authentication type is set to this value, client certificate authentication will be enforced when the following conditions are met:

- both client and server supports or requires client certificate authentication;
- the transport protocol to use is SSL;
- no credential authentication will be done.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setClientCertificateAuthentication(int)`, Constant Field Values

CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED

`public static final int` **`CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED`**

Constant indicating client certificate authentication is required. If the client certificate authentication type is set to this value, client certificate authentication will be enforced if the following conditions are met:

- both client and server supports or requires client certificate authentication;
- the transport protocol to use is SSL;
- no credential authentication will be done.

If the server doesn't support client certificate authentication and no credential authentication will be done, the client won't be able to connect to the server.

Since:
WAS XD 6.0.1

See Also:
`ClientSecurityConfiguration.setClientCertificateAuthentication(int)`, Constant Field Values

NEVER_STRING

`public static final String` **`NEVER_STRING`**

String representation for value Never indicating an option is not supported.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration settings "clientCertificateAuthentication" and "credentialAuthentication".

Since:
WAS XD 6.0.1

See Also:
`CLIENT_CERTIFICATE_AUTHENTICATION_NEVER`, `CREDENTIAL_AUTHENTICATION_NEVER`, Constant Field Values

SUPPORTED_STRING

```
public static final String SUPPORTED_STRING
```

String representation for value Supported indicating an option is supported.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration settings "clientCertificateAuthentication" and "credentialAuthentication".

Since:
WAS XD 6.0.1

See Also:
`CLIENT_CERTIFICATE_AUTHENTICATION_SUPPORTED`, `CREDENTIAL_AUTHENTICATION_SUPPORTED`, Constant Field Values

REQUIRED_STRING

```
public static final String REQUIRED_STRING
```

String representation for value Required indicating an option is required.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration settings "clientCertificateAuthentication" and "credentialAuthentication".

Since:
WAS XD 6.0.1

See Also:
`CLIENT_CERTIFICATE_AUTHENTICATION_REQUIRED`, `CREDENTIAL_AUTHENTICATION_REQUIRED`, Constant Field Values

TCPIP_STRING

```
public static final String TCPIP_STRING
```

String representation for value TCP/IP indicating a transport type of TCP/IP is used.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration setting "transportType".

Since:
WAS XD 6.0.1

See Also:
`TCP_IP`, Constant Field Values

SSL_SUPPORTED_STRING

`public static final String` **`SSL_SUPPORTED_STRING`**

String representation for value SSL-Supported indicating SSL transport type is supported.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration setting "transportType".

Since:
WAS XD 6.0.1
See Also:
`SSL_SUPPORTED`, Constant Field Values

SSL_REQUIRED_STRING

`public static final String` **`SSL_REQUIRED_STRING`**

String representation for value SSL-Required indicating SSL transport type is required.

This value is used as a value to configuration settings in a `Properties` object or property file for client and server security configurations. It is used for the configuration setting "transportType".

Since:
WAS XD 6.0.1
See Also:
`SSL_REQUIRED`, Constant Field Values

SECURE_TOKEN_MANAGER_NONE_STRING

`public static final String` **`SECURE_TOKEN_MANAGER_NONE_STRING`**

String representation for "none" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType".

Since:
WAS XD 6.0.1
See Also:
Constant Field Values

SECURE_TOKEN_MANAGER_DEFAULT_STRING

`public static final String` **`SECURE_TOKEN_MANAGER_DEFAULT_STRING`**

String representation for "default" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType". This value requires users to provide the secure token key store settings.

Since:
WAS XD 6.0.1
See Also:
Constant Field Values

SECURE_TOKEN_MANAGER_CUSTOM_STRING

```
public static final String SECURE_TOKEN_MANAGER_CUSTOM_STRING
```

String representation for "custom" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType". This value requires users to provide the SecureTokenManager implementation class name using the "customSecureTokenManagerClass" configuration setting.

Since:
WAS XD 6.0.1
See Also:
Constant Field Values

NEW_SECURE_TOKEN_MANAGER_STRING

```
public static final String NEW_SECURE_TOKEN_MANAGER_STRING
```

String representation for "autoSecret" type of the secure token manager.

This value is used in a property file for server security configurations. It is used for the configuration setting "secureTokenManagerType". This value does not require users to provide other settings.

Since:
8.6, XC10 2.5
See Also:
Constant Field Values

ACCESS_BY_CREATOR_ONLY_DISABLED

```
public static final int ACCESS_BY_CREATOR_ONLY_DISABLED
```

The access by creator only authorization is disabled.

The access by creator authorization ensures that only the user (represented by the Principals associated with it), who inserts the data entry into the map, can access the data. Here the access means read, update, invalidate, and remove.

Since:
WAS XD 6.1 FIX3
See Also:
Constant Field Values

ACCESS_BY_CREATOR_ONLY_COMPLEMENT

```
public static final int ACCESS_BY_CREATOR_ONLY_COMPLEMENT
```

The access by creator only authorization is enabled to complement the ObjectGrid map authorization.

The access by creator authorization ensures that only the user (represented by the Principals associated with it), who inserts the data entry into the map, can access the data. Here the access means read, update, invalidate, and remove.

If this constant is used, both map authorization and access by creator only authorization will take effect. Therefore, you can further limit the operations to the data entries. For example, you can restrict the creator from invalidating the data entries.

Since:
WAS XD 6.1 FIX3
See Also:
Constant Field Values

ACCESS_BY_CREATOR_ONLY_SUPERSEDE

`public static final int` **`ACCESS_BY_CREATOR_ONLY_SUPERSEDE`**

The access by creator only authorization is enabled to supersede the ObjectGrid map authorization.

The access by creator authorization ensures that only the user (represented by the Principals associated with it), who inserts the data entry into the map, can access the data. Here the access means read, update, invalidate, and remove.

If this constant is used, the access by creator only authorization will supersede the map authorization; no map authorization will be done.

Since:
WAS XD 6.1 FIX3
See Also:
Constant Field Values

Constructor Detail

SecurityConstants

`public` **`SecurityConstants`**`()`

com.ibm.websphere.objectgrid.security

Class ObjectGridSecurityException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.security.ObjectGridSecurityException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

CannotGenerateCredentialException, ExpiredCredentialException, InvalidCredentialException

---

```
public class ObjectGridSecurityException
extends ObjectGridException
```

This exception represents a general ObjectGrid security exception.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridSecurityException() Constructs a new ObjectGridSecurityException with `null` as its detail message.
ObjectGridSecurityException(String message) Constructs a new ObjectGridSecurityException with the specified detail message.
ObjectGridSecurityException(String message, Throwable cause) Constructs a new ObjectGridSecurityException with the specified detail message and cause.
ObjectGridSecurityException(Throwable cause) Constructs a new ObjectGridSecurityException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridSecurityException

public **ObjectGridSecurityException**()

Constructs a new ObjectGridSecurityException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

ObjectGridSecurityException

public **ObjectGridSecurityException**(String message)

Constructs a new ObjectGridSecurityException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
ObjectGridException.initCause(Throwable), Throwable.getMessage()

ObjectGridSecurityException

public **ObjectGridSecurityException**(String message,
Throwable cause)

Constructs a new ObjectGridSecurityException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridSecurityException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
ObjectGridException.getCause(), Throwable.getMessage()

ObjectGridSecurityException

public **ObjectGridSecurityException**(Throwable cause)

Constructs a new ObjectGridSecurityException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for

ObjectGridSecurityExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

Package com.ibm.websphere.objectgrid.spring

This package holds the Spring specific APIs for ObjectGrid.

See:

Description

Interface Summary

SpringLocalTxManager	This interface has the methods for interacting with the ObjectGrid Spring LocalTransactionManager.

Class Summary

ObjectGridCache	This class is a WebSphere eXtreme Scale implementation of the Spring Framework's Cache interface.
ObjectGridCatalogServiceDomainBean	This class is a Spring bean representing a Catalog Service Domain for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.
ObjectGridClientBean	This class is a Spring bean representing an ObjectGrid client instance for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.
ObjectGridSpringFactory	This class serves as a factory to construct instances of the various Spring specific APIs.

Exception Summary

CannotGetObjectGridSessionException	This can be thrown by getSession if a new Session cannot be obtained for any reason.
ObjectGridTransactionException	The ObjectGrid platform transaction manager can throw this unchecked exception whenever errors occur.

Package com.ibm.websphere.objectgrid.spring Description

This package holds the Spring specific APIs for ObjectGrid.

Local Transaction Support

ObjectGrid has implemented a Spring PlatformTransactionManager. This allows Spring to manage local transactions using a single ObjectGrid session. Spring can then be used to annotate POJOs with container managed transaction semantics much like a J2EE application server does using J2EE CMT. An application should instantiate a SpringLocalTxManager using the appropriate factory method on ObjectGridSpringFactory and then wire a reference to that object in to all POJOs that use Spring CMT. This instance has a getSession method to obtain the correct Session for that POJO. The application must call one of the SpringLocalTxManager#setObjectGridForThread methods before invoking a managed POJO to specify which ObjectGrid instance should be used for any CMT on this thread.

---

com.ibm.websphere.objectgrid.spring

Interface SpringLocalTxManager

public interface **SpringLocalTxManager**

This interface has the methods for interacting with the ObjectGrid Spring LocalTransactionManager. It also allows the desired partition to use with this thread to be specified.

Since:
WAS XD 6.1 FIX3, XC10

See Also:
ObjectGridSpringFactory.getLocalPlatformTransactionManager()

Method Summary

Session	**getSession**() This returns a managed session for the ObjectGrid associated with this thread.
void	**setObjectGridForThread**(ObjectGrid grid) This indicates the ObjectGrid to use on this thread when a session is requested.

Method Detail

setObjectGridForThread

void **setObjectGridForThread**(ObjectGrid grid)

This indicates the ObjectGrid to use on this thread when a session is requested. This replaces any previously associated ObjectGrid, i.e. only a single grid instance can be associated with a thread at a time.

Parameters:
grid - the ObjectGrid to set on this thread.

See Also:
getSession()

getSession

Session **getSession**()

This returns a managed session for the ObjectGrid associated with this thread.

Do not call `begin()`, `commit()` or `rollback()` directly on the session. Spring manages the transaction automatically.

Returns:

A managed Session to use with this thread.

Throws:

`CannotGetObjectGridSessionException` - thrown when an ObjectGrid session can't be retrieved.

See Also:

`setObjectGridForThread(ObjectGrid)`

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridTransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        org.springframework.core.NestedRuntimeException
          org.springframework.transaction.TransactionException
            com.ibm.websphere.objectgrid.spring.ObjectGridTransactionException

All Implemented Interfaces:

Serializable

---

public class **ObjectGridTransactionException**
extends TransactionException

The ObjectGrid platform transaction manager can throw this unchecked exception whenever errors occur.

Since:

WAS XD 6.1 FIX3, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridTransactionException(String message) Constructs a new ObjectGridTransactionException with the specified detail message.
ObjectGridTransactionException(String message, Throwable cause) Constructs a new ObjectGridTransactionException with the specified detail message and cause.

Method Summary

Methods inherited from class org.springframework.core.NestedRuntimeException
contains, getMessage, getMostSpecificCause, getRootCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridTransactionException

```
public ObjectGridTransactionException(String message,
                                      Throwable cause)
```

Constructs a new ObjectGridTransactionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridTransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`Throwable.getCause()`, `NestedRuntimeException.getMessage()`

ObjectGridTransactionException

```
public ObjectGridTransactionException(String message)
```

Constructs a new ObjectGridTransactionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`Throwable.initCause(Throwable)`, `NestedRuntimeException.getMessage()`

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridSpringFactory

java.lang.Object

↳ **com.ibm.websphere.objectgrid.spring.ObjectGridSpringFactory**

```
public final class ObjectGridSpringFactory
extends Object
```

This class serves as a factory to construct instances of the various Spring specific APIs.

Since:

WAS XD 6.1 FIX3, XC10

Field Summary

static String	**SCOPE_SHARD** Scope identifier for shard scope: "shard".

Constructor Summary

ObjectGridSpringFactory()

Method Summary

static BeanFactory	**getBeanFactoryForObjectGrid**(String objectGridName) This returns the currently registered external bean factory for a named object grid.
sta	

static Object	**getBeanInShardScope**(ObjectGrid shard, String beanName) This returns an instance of the named Spring bean with the current shard scope using the specified ObjectGrid instance.
static SpringLocalTxManager	**getLocalPlatformTransactionManager**() This returns an ObjectGrid PlatformLocalTransactionManager.
static void	**registerSpringBeanFactoryAdapter**(String objectGridName, Object springBeanFactory) This returns an adapter for a Spring based bean factory.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SCOPE_SHARD

public static final String **SCOPE_SHARD**

Scope identifier for shard scope: "shard".

See Also:

getBeanInShardScope(ObjectGrid, String), Constant Field Values

Constructor Detail

ObjectGridSpringFactory

public **ObjectGridSpringFactory**()

Method Detail

getLocalPlatformTransactionManager

```
public static SpringLocalTxManager getLocalPlatformTransactionManager()
```

This returns an ObjectGrid PlatformLocalTransactionManager.

Returns:
the PlatformLocalTransactionManager instance.

---

registerSpringBeanFactoryAdapter

```
public static void registerSpringBeanFactoryAdapter(String objectGridName,
                                                    Object springBeanFactory)
                                             throws ClassCastException
```

This returns an adapter for a Spring based bean factory. We use an Object type here to avoid making ObjectGrid dependent on Spring classes being present. A ClassCastException exception is thrown if the supplied factory isn't a Spring BeanFactory instance.

Parameters:
`objectGridName` - the name of the ObjectGrid
`springBeanFactory` - A Spring BeanFactory instance.
Throws:
`ClassCastException` - thrown when the Object type is not a BeanFactory instance.

---

getBeanFactoryForObjectGrid

```
public static BeanFactory getBeanFactoryForObjectGrid(String objectGridName)
```

This returns the currently registered external bean factory for a named object grid. If no factory has been registered then it attempts to construct a Spring BeanFactory using the xml resource on the class path @ "/X_spring.xml" and /META-INF/X_spring.xml where X is the name of the ObjectGrid. If the xml file is on the class path then the ObjectGrid name MUST be a valid resource name.

Parameters:
`objectGridName` - The name of the ObjectGrid
Returns:
The BeanFactory instance or null if there were none registered

---

getBeanInShardScope

```
public static Object getBeanInShardScope(ObjectGrid shard,
                                         String beanName)
```

This returns an instance of the named Spring bean with the current shard scope using the specified ObjectGrid instance. This allows shard scoped beans to be obtained.

Parameters:
`shard` - The ObjectGrid instance to use to scope Spring beans using "shard" as scope.
`beanName` - The bean to return
Returns:
The bean instance if it exists
See Also:
`SCOPE_SHARD`

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

com.ibm.websphere.objectgrid.spring

Class ObjectGridClientBean

java.lang.Object
  com.ibm.websphere.objectgrid.spring.ObjectGridClientBean

All Implemented Interfaces:

InitializingBean

```
public final class ObjectGridClientBean
extends Object
implements InitializingBean
```

This class is a Spring bean representing an ObjectGrid client instance for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.

Users must provide a ObjectGridCatalogServiceDomainBean to configure this client. The ObjectGrid name is optional when using the provided XML configuration files.

Since:

8.5, XC10

Constructor Summary

ObjectGridClientBean()

Method Summary

void	**afterPropertiesSet**() Initializes the client bean.
void	**setCatalogServiceDomain**(ObjectGridCatalogServiceDomainBean catalogServiceDomain) Sets the ObjectGridCatalogServiceDomainBean used by the client.
void	**setObjectGridName**(String objectGridName) Sets the name of the ObjectGrid for the client.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridClientBean

`public` **`ObjectGridClientBean`**`()`

Method Detail

setObjectGridName

`public void` **`setObjectGridName`**`(`String `objectGridName)`

Sets the name of the ObjectGrid for the client. This is optional when using the provided XML configuration files.

Parameters:

`objectGridName` - The name of the ObjectGrid to connect to.

---

setCatalogServiceDomain

`public void` **`setCatalogServiceDomain`**`(`ObjectGridCatalogServiceDomainBean `catalogServiceDomain)`

Sets the ObjectGridCatalogServiceDomainBean used by the client.

Parameters:

`catalogServiceDomain` - The ObjectGridCatalogServiceDomainBean for the client.

---

afterPropertiesSet

`public void` **`afterPropertiesSet`**`()`
`throws` Exception

Initializes the client bean. This method is to be called after all setter methods have been called and will throw an IllegalArgumentException if setCatalogServiceDomain(ObjectGridCatalogServiceDomainBean) has not been called prior.

Specified by:

afterPropertiesSet in interface InitializingBean

Throws:

Exception - if setCatalogServiceDomain(ObjectGridCatalogServiceDomainBean) has not been called prior or an error occurs retrieving the ObjectGrid.

See Also:

InitializingBean.afterPropertiesSet()

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridCatalogServiceDomainBean

java.lang.Object
  com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainBean

All Implemented Interfaces:

InitializingBean

```
public final class ObjectGridCatalogServiceDomainBean
extends Object
implements InitializingBean
```

This class is a Spring bean representing a Catalog Service Domain for use with the WebSphere eXtreme Scale implementation of Spring's cache abstraction.

Users must provide the catalog service endpoints used to connect to the eXtreme Scale cluster/domain. Users optionally may provide a client override XML and/or a client security configuration.

Since:

8.5, XC10

Constructor Summary

ObjectGridCatalogServiceDomainBean()

Method Summary

void	**afterPropertiesSet**() Initializes the connection to the Catalog Service Domain.
void	**setCatalogServiceEndpoints**(String catalogServiceEndpoints) Sets the catalog service endpoints used to connect to the cluster/domain.
void	**setClientOverrideXml**(Resource clientOverrideXml) Sets the location of the client override XML.
void	**setClientSecurityConfig**(Resource clientSecurityConfiguration) Sets the location of the client security configuration.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridCatalogServiceDomainBean

public **ObjectGridCatalogServiceDomainBean**()

Method Detail

setCatalogServiceEndpoints

public void **setCatalogServiceEndpoints**(String catalogServiceEndpoints)

Sets the catalog service endpoints used to connect to the cluster/domain.

Parameters:

catalogServiceEndpoints - The catalog service endpoints to connect to the cluster/domain

---

setClientOverrideXml

public void **setClientOverrideXml**(Resource clientOverrideXml)

Sets the location of the client override XML.

Parameters:

clientOverrideXml - The location of the client override XML.

---

setClientSecurityConfig

public void **setClientSecurityConfig**(Resource clientSecurityConfiguration)

Sets the location of the client security configuration.

Parameters:

clientSecurityConfiguration - The location of the client security configuration.

---

afterPropertiesSet

public void **afterPropertiesSet**()
throws Exception

Initializes the connection to the Catalog Service Domain. This method is to be called after all setter methods have been called and will throw an IllegalArgumentException if setCatalogServiceEndpoints(String) have not been called prior.

Specified by:

afterPropertiesSet in interface InitializingBean

Throws:

Exception - if setCatalogServiceEndpoints(String) have not been called prior or an error occurs initializing the connection.

See Also:

InitializingBean.afterPropertiesSet()

---

com.ibm.websphere.objectgrid.spring

Class ObjectGridCache

java.lang.Object
  com.ibm.websphere.objectgrid.spring.ObjectGridCache

All Implemented Interfaces:
InitializingBean, Cache

```
public final class ObjectGridCache
extends Object
implements Cache, InitializingBean
```

This class is a WebSphere eXtreme Scale implementation of the Spring Framework's Cache interface.

Users must provide a name and a ObjectGridClientBean to configure this cache. The ObjectMap name is optional when using the provided XML configuration files.

This implementation allows for the storage of null values and does not support null keys.

The following Spring Inversion of Control (IoC) container configuration snippet creates two caches, named default and books hosted by the catalog service domain with connection end-points of host1:2809,host2:2809.

```
 <bean id="wxsCSDomain" class="com.ibm.websphere.objectgrid.spring.ObjectGridCatalogServiceDomainB
ean"
     p:catalog-service-endpoints="host1:2809,host2:2809" />

 <bean id="wxsGridClient" class="com.ibm.websphere.objectgrid.spring.ObjectGridClientBean"
      p:catalog-service-domain-ref="wxsCSDomain" />

 <bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">
     <property name="caches">
         <set>
             <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
                 p:name="default"
                 p:object-grid-client-ref="wxsGridClient" />
             <bean class="com.ibm.websphere.objectgrid.spring.ObjectGridCache"
                 p:name="books"
                 p:object-grid-client-ref="wxsGridClient" />
         </set>
     </property>
 </bean>
```

Since:
8.5, XC10

Nested Class Summary

Nested classes/interfaces inherited from interface

org.springframework.cache.Cache
Cache.ValueWrapper

Constructor Summary

ObjectGridCache()

Method Summary

void	**afterPropertiesSet**() Initializes this cache.
void	**clear**() Clears all entries from the cache.
void	**evict**(Object key) Evicts the entry at the given key.
Cache.ValueWrapper	**get**(Object key) Retrieves the object from the cache at the given key.
String	**getName**() Returns the name of the cache.
Object	**getNativeCache**() This implementation returns `null`.
void	**put**(Object key, Object value) Creates an entry in the cache.
void	**setMapName**(String mapName) Sets the ObjectMap name.

void	**setName**(String name) Sets the cache name.
void	**setObjectGridClient**(ObjectGridClientBean objectGridClient) Sets the ObjectGridClientBean to use.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridCache

public **ObjectGridCache**()

Method Detail

afterPropertiesSet

public void **afterPropertiesSet**()
throws Exception

Initializes this cache. This method is to be called after all setter methods have been called and will throw an IllegalArgumentException if setObjectGridClient(ObjectGridClientBean) or setName(String) have not been called prior.

Specified by:
afterPropertiesSet in interface InitializingBean

Throws:
Exception - if setObjectGridClient(ObjectGridClientBean) or setName(String) have not been called prior or an error occurs initializing the cache.

See Also:
InitializingBean.afterPropertiesSet()

---

getName

public String **getName**()

Returns the name of the cache.

Specified by:
getName in interface Cache

Returns:
The name of the cache.

See Also:
Cache.getName()

---

setName

public void **setName**(String name)

Sets the cache name.

Parameters:

`name` - The cache name.

See Also:

`getName()`

---

setMapName

```
public void setMapName(String mapName)
```

Sets the ObjectMap name. This is optional when using the provided XML configuration files.

Parameters:

`mapName` - The name of the ObjectMap

---

setObjectGridClient

```
public void setObjectGridClient(ObjectGridClientBean objectGridClient)
```

Sets the `ObjectGridClientBean` to use.

Parameters:

`objectGridClient` - The `ObjectGridClientBean` to use

---

getNativeCache

```
public Object getNativeCache()
```

This implementation returns `null`.

Specified by:

`getNativeCache` in interface `Cache`

Returns:

This implementation returns `null`.

See Also:

`Cache.getNativeCache()`

---

get

```
public Cache.ValueWrapper get(Object key)
```

Retrieves the object from the cache at the given key. Returns `null` if there is no mapping for the key or if the key is `null`

Specified by:

`get` in interface `Cache`

Returns:

the object from the cache at the given key or `null` if there is no mapping or the key is `null`

See Also:

`Cache.get(java.lang.Object)`

---

put

```
public void put(Object key,
                Object value)
```

Creates an entry in the cache. Overwrites the value if an entry for the given key exists. This method does not create an entry if the given key is `null`.

Specified by:
`put` in interface `Cache`

See Also:
`Cache.put(java.lang.Object, java.lang.Object)`

---

evict

`public void **evict**(Object key)`

Evicts the entry at the given key. If the provided key is `null` this method does not evict any entries.

Specified by:
`evict` in interface `Cache`

See Also:
`Cache.evict(java.lang.Object)`

---

clear

`public void **clear**()`

Clears all entries from the cache.

Specified by:
`clear` in interface `Cache`

See Also:
`Cache.clear()`

---

com.ibm.websphere.objectgrid.spring

Class CannotGetObjectGridSessionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        org.springframework.core.NestedRuntimeException
          com.ibm.websphere.objectgrid.spring.CannotGetObjectGridSessionException

All Implemented Interfaces:

Serializable

```
public class CannotGetObjectGridSessionException
extends NestedRuntimeException
```

This can be thrown by getSession if a new Session cannot be obtained for any reason.

Since:

WAS XD 6.1 FIX3, XC10

See Also:

Serialized Form

Constructor Summary

CannotGetObjectGridSessionException(String message) Constructs a new CannotGetObjectGridSessionException with the specified detail message.
CannotGetObjectGridSessionException(String message, Throwable cause) Constructs a new CannotGetObjectGridSessionException with the specified detail message and cause.

Method Summary

Methods inherited from class org.springframework.core.NestedRuntimeException
contains, getMessage, getMostSpecificCause, getRootCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CannotGetObjectGridSessionException

```
public CannotGetObjectGridSessionException(String message,
                                           Throwable cause)
```

Constructs a new CannotGetObjectGridSessionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this CannotGetObjectGridSessionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`Throwable.getCause()`, `NestedRuntimeException.getMessage()`

CannotGetObjectGridSessionException

```
public CannotGetObjectGridSessionException(String message)
```

Constructs a new CannotGetObjectGridSessionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`Throwable.initCause(Throwable)`, `NestedRuntimeException.getMessage()`

---

com.ibm.websphere.objectgrid

Class UndefinedMapException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.UndefinedMapException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class UndefinedMapException
extends ObjectGridException
```

This exception indicates that the map which an application tries to access is not defined in the ObjectGrid.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

UndefinedMapException() Constructs a new UndefinedMapException with `null` as its detail message.
UndefinedMapException(String message) Constructs a new UndefinedMapException with the specified detail message.
UndefinedMapException(String message, Throwable cause) Constructs a new UndefinedMapException with the specified detail message and cause.
UndefinedMapException(Throwable cause) Constructs a new UndefinedMapException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

UndefinedMapException

public **UndefinedMapException**()

Constructs a new UndefinedMapException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

UndefinedMapException

public **UndefinedMapException**(String message)

Constructs a new UndefinedMapException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

UndefinedMapException

public **UndefinedMapException**(String message,
Throwable cause)

Constructs a new UndefinedMapException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this UndefinedMapException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

UndefinedMapException

public **UndefinedMapException**(Throwable cause)

Constructs a new UndefinedMapException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for UndefinedMapExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that

the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class UnavailableServiceException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException
          com.ibm.websphere.objectgrid.UnavailableServiceException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

TransactionAffinityException, TransactionQuiesceException

```
public class UnavailableServiceException
extends LoaderException
```

This exception is thrown when all servers are dead or when all services are unavailable even though servers are running.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

UnavailableServiceException() Constructs a new UnavailableServiceException with `null` as its detail message.
UnavailableServiceException(String message) Constructs a new UnavailableServiceException with the specified detail message.
UnavailableServiceException(String message, Throwable cause) Constructs a new UnavailableServiceException with the specified detail message and cause.
UnavailableServiceException(Throwable cause) Constructs a new UnavailableServiceException with a specified cause.

Method Summary

int	**getReplicationGroup**() Returns the replication group identifier for this exception.
void	**setReplicationGroup**(int replicationGroup) Sets the replication group identifier for this exception.

d	

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

UnavailableServiceException

```
public UnavailableServiceException()
```

Constructs a new UnavailableServiceException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

UnavailableServiceException

```
public UnavailableServiceException(String message)
```

Constructs a new UnavailableServiceException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

UnavailableServiceException

```
public UnavailableServiceException(String message,
                                   Throwable cause)
```

Constructs a new UnavailableServiceException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this UnavailableServiceException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

UnavailableServiceException

`public` **`UnavailableServiceException`**`(`[`Throwable`]` cause)`

Constructs a new UnavailableServiceException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for UnavailableServiceExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

[`ObjectGridException.getCause()`]

Method Detail

getReplicationGroup

`public int` **`getReplicationGroup`**`()`

Returns the replication group identifier for this exception.

Returns:

the argument that was passed to the `setReplicationGroup(int)` method of this class or `0` if the `setReplicationGroup` method was not previously called for this object.

See Also:

[`setReplicationGroup(int)`]

setReplicationGroup

`public void` **`setReplicationGroup`**`(int replicationGroup)`

Sets the replication group identifier for this exception.

Parameters:

`replicationGroup` - The replication group identifier

com.ibm.websphere.objectgrid

Interface TxID

All Superinterfaces:

Serializable

```
public interface TxID
extends Serializable
```

This interface is an opaque identifier for a transaction. Context information can be stored and retrieved in multiple slots on this object. This mechanism allows a `TransactionCallback` and `Loader`, for example, to share state information with each other in the context of a specific session transaction.

The TxID.toString() output can be used to determine whether the originating Session transaction is a single partition transaction or a multi-partition transaction. If the String output begins with the keyword `Local` then this indicates a single partition transaction, for example: `Local-40000139-72B2-C037-E000-1C271366B073`

If the String output begins with the keyword `WXS` then this indicates a multi-partition transaction, for example: `WXS-40000139-72B2-BD3A-E000-1C271366B073`

Since:

WAS XD 6.0, XC10

See Also:

`Loader`, `ObjectGrid.reserveSlot(String)`, `Session`, `TransactionCallback`

Field Summary

static String	**SLOT_NAME** All slots should be reserved using this name.

Method Summary

boolean	**equals**(TxID o) Checks for equality between two TxID objects.
Se	

Session	**getSession**() Returns the Session that owns this TxID.
Object	**getSlot**(int slotNumber) Gets the context information currently associated with this transaction.
int	**hashCode**() Returns the hashcode of the Tx identifier.
void	**putSlot**(int slotNumber, Object o) Sets some context information to be associated with this transaction.

Field Detail

SLOT_NAME

static final String **SLOT_NAME**

All slots should be reserved using this name.

See Also:
ObjectGrid.reserveSlot(String), Constant Field Values

Method Detail

equals

boolean **equals**(TxID o)

Checks for equality between two TxID objects.

Parameters:
o - Input TxID to check for equality against
Returns:
true, if they are equal; false, if they not equal

---

hashCode

int **hashCode**()

Returns the hashcode of the Tx identifier.

Overrides:
hashCode in class Object

Returns:
hashcode

---

getSlot

`Object` **`getSlot`**`(int slotNumber)`

Gets the context information currently associated with this transaction.

Parameters:
`slotNumber` - the slot number for the context information being requested
Returns:
Object the current context information for the slot number
See Also:
`putSlot(int, Object)`, `ObjectGrid.reserveSlot(String)`

---

putSlot

`void` **`putSlot`**`(int slotNumber,`
`Object o)`

Sets some context information to be associated with this transaction.

Parameters:
`slotNumber` - the slot number
`o` - Object to be put into the TxID slot
See Also:
`getSlot(int)`, `ObjectGrid.reserveSlot(String)`

---

getSession

`Session` **`getSession`**`()`

Returns the Session that owns this TxID.

Returns:
a Session object to use.
See Also:
`Session`

---

com.ibm.websphere.objectgrid

Class TransactionTimeoutException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        com.ibm.websphere.objectgrid.ObjectGridRuntimeException
          com.ibm.websphere.objectgrid.TransactionTimeoutException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionTimeoutException**
extends ObjectGridRuntimeException

This exception is thrown when a transaction exceeds the transaction timeout that was specified on the `ObjectGrid` or `Session`.

Since:

WAS XD 6.0.1, XC10

See Also:

ObjectGrid.setTxTimeout(int), Session.setTransactionTimeout(int), Serialized Form

---

Constructor Summary

TransactionTimeoutException(String message, String txIdString) Constructs a new TransactionTimeoutException with the specified detail message.
TransactionTimeoutException(String message, Throwable cause)

Method Summary

String	**getTxIDString**() Get the String representation of the `TxID` for the transaction that timed out.
Date	**whenOccurred**() Gives the time when this TransactionTimeoutException was created.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridRuntimeException

getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionTimeoutException

```
public TransactionTimeoutException(String message,
                                   Throwable cause)
```

TransactionTimeoutException

```
public TransactionTimeoutException(String message,
                                   String txIdString)
```

Constructs a new TransactionTimeoutException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`txIdString` - the result of `TxID.toString()` for the transaction that timed out.

See Also:

ObjectGridRuntimeException.initCause(Throwable), Throwable.getMessage()

Method Detail

whenOccurred

```
public Date whenOccurred()
```

Gives the time when this TransactionTimeoutException was created.

Returns:

Date object that represents the instant in time when this exception object was created.

---

getTxIDString

```
public String getTxIDString()
```

Get the String representation of the `TxID` for the transaction that timed out.

Returns:

String value of TxID of transaction that timed out.

---

com.ibm.websphere.objectgrid

Class TransactionQuiesceException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException
          com.ibm.websphere.objectgrid.UnavailableServiceException
            com.ibm.websphere.objectgrid.TransactionQuiesceException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionQuiesceException**
extends UnavailableServiceException

This exception is thrown when partition/shard/mapset/replication group/ replication group member/server/cluster/objectgrid is entered quiesce process for various reasons such as shard movement, partition relocation, system update, server shutdown, and others. Quiesce process ensures data integrity and transaction integrity. This exception is thrown only for new start of a new transaction; it will not impact old transaction requests that are allowed to finish.

Since:

WAS XD 6.1, XC10

See Also:

Serialized Form

Constructor Summary

TransactionQuiesceException() Constructs a new TransactionQuiesceException with `null` as its detail message.
TransactionQuiesceException(String message) Constructs a new TransactionQuiesceException with the specified detail message.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.UnavailableServiceException

getReplicationGroup, setReplicationGroup

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionQuiesceException

public **TransactionQuiesceException**()

Constructs a new TransactionQuiesceException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

TransactionQuiesceException

public **TransactionQuiesceException**(String message)

Constructs a new TransactionQuiesceException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

---

com.ibm.websphere.objectgrid

Class TransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.TransactionException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

SessionNotReentrantException, TransactionAlreadyActiveException

---

```
public class TransactionException
extends ObjectGridException
```

A general transaction exception indicating something went wrong with a transaction. The `isTransactionActive()` and `wasTransactionRolledBack()` methods can be used to determine whether transaction is still active or was rolled back as a result of this exception occuring.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Field Summary

protected boolean	**ivTransactionRolledBack** Indicates whether the transaction was rolled back or not.

Constructor Summary

TransactionException(String message, boolean rolledBack) Constructs a new TransactionException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception.

TransactionException(String message, Throwable cause, boolean rolledBack) Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String message, TransactionException cause, boolean rolledBack) Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.
TransactionException(Throwable cause, boolean rolledBack) Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception.
TransactionException(TransactionException cause, boolean rolledBack) Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception.

Method Summary

boolean	**isTransactionActive**() Returns true if the transaction is active.
boolean	**wasTransactionRolledBack**() Returns true if the transaction was rolled back.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ivTransactionRolledBack

```
protected boolean ivTransactionRolledBack
```

Indicates whether the transaction was rolled back or not.

Constructor Detail

TransactionException

```
public TransactionException(String message,
                            boolean rolledBack)
```

Constructs a new TransactionException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(Throwable cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for as a wrapper for other Throwable objects that occur.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

`ObjectGridException.getCause()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(TransactionException cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with a specified cause and a specified indication of whether the transaction was rolled back as a result of this exception. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for as a wrapper for other Throwable objects that occur.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

Since:

WAS XD 6.1 IFIX1

See Also:

`ObjectGridException.getCause()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(String message,
                            Throwable cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`, `wasTransactionRolledBack()`

TransactionException

```
public TransactionException(String message,
                            TransactionException cause,
                            boolean rolledBack)
```

Constructs a new TransactionException with the specified detail message, cause, and indication of whether the transaction was rolled back as a result of this exception.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
`rolledBack` - A value of `true` indicates the transaction was rolled back.

Since:

WAS XD 6.1 IFIX1

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`, `wasTransactionRolledBack()`

Method Detail

isTransactionActive

```
public boolean isTransactionActive()
```

Returns `true` if the transaction is active. Otherwise, `false` is returned to indicate either the transaction never started or was completed.

Returns:

true if the transaction is active

---

wasTransactionRolledBack

```
public boolean wasTransactionRolledBack()
```

Returns `true` if the transaction was rolled back.

Returns:

true if the transaction was rolled back

---

com.ibm.websphere.objectgrid

Class TransactionAlreadyActiveException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.TransactionException
          com.ibm.websphere.objectgrid.TransactionAlreadyActiveException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionAlreadyActiveException**
extends TransactionException

An exception indicating a transaction is already active for the current session. This exception does not cause the current active transaction to be rolled back, so the `isTransactionActive` method will return `true`.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Field Summary

Fields inherited from class com.ibm.websphere.objectgrid.TransactionException
ivTransactionRolledBack

Constructor Summary

TransactionAlreadyActiveException() Constructs a new TransactionAlreadyActiveException with `null` as its detail message.
TransactionAlreadyActiveException(String message) Constructs a new TransactionAlreadyActiveException with the specified detail message.
TransactionAlreadyActiveException(String message, Throwable cause) Constructs a new TransactionAlreadyActiveException with the specified detail message and cause.
TransactionAlreadyActiveException(Throwable cause) Constructs a new TransactionAlreadyActiveException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.TransactionException
isTransactionActive, wasTransactionRolledBack

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionAlreadyActiveException

public **TransactionAlreadyActiveException**()

Constructs a new TransactionAlreadyActiveException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

TransactionAlreadyActiveException

public **TransactionAlreadyActiveException**(String message)

Constructs a new TransactionAlreadyActiveException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

TransactionAlreadyActiveException

public **TransactionAlreadyActiveException**(Throwable cause)

Constructs a new TransactionAlreadyActiveException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for TransactionAlreadyActiveExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:
ObjectGridException.getCause()

TransactionAlreadyActiveException

```
public TransactionAlreadyActiveException(String message,
                                         Throwable cause)
```

Constructs a new TransactionAlreadyActiveException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionAlreadyActiveException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
ObjectGridException.getCause(), Throwable.getMessage()

---

com.ibm.websphere.objectgrid

Class TransactionAffinityException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.LoaderException
          com.ibm.websphere.objectgrid.UnavailableServiceException
            com.ibm.websphere.objectgrid.TransactionAffinityException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **TransactionAffinityException**
extends UnavailableServiceException

This exception is thrown for inflight transaction when server fails over. We suggest applications to retry transaction.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

TransactionAffinityException() Constructs a new TransactionAffinityException with `null` as its detail message.
TransactionAffinityException(String message) Constructs a new TransactionAffinityException with the specified detail message.
TransactionAffinityException(String message, Throwable cause) Constructs a new TransactionAffinityException with the specified detail message and cause.
TransactionAffinityException(Throwable cause) Constructs a new TransactionAffinityException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.UnavailableServiceException
getReplicationGroup, setReplicationGroup

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionAffinityException

`public` **`TransactionAffinityException`**`()`

Constructs a new TransactionAffinityException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

TransactionAffinityException

`public` **`TransactionAffinityException`**`(String message)`

Constructs a new TransactionAffinityException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

TransactionAffinityException

`public` **`TransactionAffinityException`**`(String message,`
`Throwable cause)`

Constructs a new TransactionAffinityException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TransactionAffinityException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

TransactionAffinityException

`public` **`TransactionAffinityException`**`(Throwable cause)`

Constructs a new TransactionAffinityException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for TransactionAffinityExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class TargetNotAvailableException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        com.ibm.websphere.objectgrid.TargetNotAvailableException

All Implemented Interfaces:

Serializable

---

public class **TargetNotAvailableException**
extends RuntimeException

A `TargetNotAvailableException` indicates the ObjectGrid target is not available. This could be due to the fact that ObjectGrid servers are not available or the ObjectGrid placement has not finished.

Since:

WAS XD 6.1, XC10

See Also:

Serialized Form

Constructor Summary

TargetNotAvailableException() Constructs a new TargetNotAvailableException with `null` as its detail message.
TargetNotAvailableException(String message) Constructs a new TargetNotAvailableException with the specified detail message.
TargetNotAvailableException(String message, Throwable cause) Constructs a new TargetNotAvailableException with the specified detail message and cause.
TargetNotAvailableException(Throwable cause) Constructs a new TargetNotAvailableException with a specified cause.

Method Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

---

Constructor Detail

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`()`

Constructs a new TargetNotAvailableException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

`Throwable.initCause(Throwable)`

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`(String message)`

Constructs a new TargetNotAvailableException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`Throwable.initCause(Throwable)`, `Throwable.getMessage()`

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`(Throwable cause)`

Constructs a new TargetNotAvailableException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for TargetNotAvailableExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`Throwable.getCause()`

TargetNotAvailableException

`public` **`TargetNotAvailableException`**`(String message,`
`Throwable cause)`

Constructs a new TargetNotAvailableException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this TargetNotAvailableException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

Throwable.getCause(), Throwable.getMessage()

---

com.ibm.websphere.objectgrid

Class TTLType

java.lang.Object

  com.ibm.websphere.objectgrid.TTLType

All Implemented Interfaces:

Serializable

```
public class TTLType
extends Object
implements Serializable
```

Every BackingMap in ObjectGrid has a built in timed based evictor that is referred to as "time to live" evictor or TTL evictor. Each BackingMap entry has an expiration time that determines how long the entry is allowed to live in the BackingMap. When the expiration time is reached, the TTL evictor causes the expired entry to be evicted from the BackingMap. This class is used to define the TTLType value constants that determine how the the expiration time is computed for a map entry.

Since:

WAS XD 6.0, XC10

See Also:

BackingMap.setTtlEvictorType(TTLType), Serialized Form

Field Summary

static TTLType	**CREATION_TIME** A TTLType.CREATION_TIME indicates an entry expiration time is the sum of the creation time of the entry plus the "time to live" value.
static TTLType	**LAST_ACCESS_TIME** A TTLType.LAST_ACCESS_TIME indicates an entry expiration time is the sum of the last access time of the entry plus the "time to live" value.
sta	

static TTLType	**LAST_UPDATE_TIME** A TTLType.LAST_UPDATE_TIME indicates an entry expiration time is the sum of the last update time of the entry plus the "time to live" value.
static TTLType	**NONE** A TTLType.NONE indicates an entry has no expiration time and is allowed to live in the BackingMap until the application explicitly removes or invalidates the entry or a user defined evictor evicts it.

Method Summary

byte	**getId**() Get the raw value of this TTLType.
String	**toString**() Returns a string representation of the `TTLType`.
static TTLType	**valueOf**(byte id) Given the raw value of a TTLType, this method returns a TTLType object, or null if the raw value does not match an existing type.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

NONE

```
public static final TTLType NONE
```

A TTLType.NONE indicates an entry has no expiration time and is allowed to live in the BackingMap until the application explicitly removes or invalidates the entry or a user defined evictor evicts it.

CREATION_TIME

`public static final TTLType` **`CREATION_TIME`**

A TTLType.CREATION_TIME indicates an entry expiration time is the sum of the creation time of the entry plus the "time to live" value. The "time to live" value is set using the `BackingMap.setTimeToLive(int)` method and is the same for every entry and can **not** be changed by the application by using the `ObjectMap.setTimeToLive(int)` method. It can only be set prior to ObjectGrid initialization by use of the `BackingMap.setTimeToLive(int)` method.

See Also:

`BackingMap.setTimeToLive(int)`

LAST_ACCESS_TIME

`public static final TTLType` **`LAST_ACCESS_TIME`**

A TTLType.LAST_ACCESS_TIME indicates an entry expiration time is the sum of the last access time of the entry plus the "time to live" value. By default, the time to live value is set using the `BackingMap.setTimeToLive(int)` method and the default can be overridden by the application by using the `ObjectMap.setTimeToLive(int)` method.

See Also:

`BackingMap.setTimeToLive(int)`, `ObjectMap.setTimeToLive(int)`

LAST_UPDATE_TIME

`public static final TTLType` **`LAST_UPDATE_TIME`**

A TTLType.LAST_UPDATE_TIME indicates an entry expiration time is the sum of the last update time of the entry plus the "time to live" value. By default, the time to live value is set using the `BackingMap.setTimeToLive(int)` method and the default can be overridden by the application by using the `ObjectMap.setTimeToLive(int)` method. The difference between this TTLType and LAST_ACCESS_TIME is that fetch operations do not cause the entry expiration time to be updated.

Since:

7.1

See Also:

`BackingMap.setTimeToLive(int)`, `ObjectMap.setTimeToLive(int)`

Method Detail

valueOf

`public static final TTLType` **`valueOf`**`(byte id)`

Given the raw value of a TTLType, this method returns a TTLType object, or null if the raw value does not match an existing type. This method is used to deserialize this object.

Parameters:

`id` - the raw value of a TTLType

Returns:

the TTLType corresponding to the raw input value

Since:

8.6, XC10 2.5

---

getId

`public byte **getId**()`

Get the raw value of this TTLType. This method is used to serialize this object.

Returns:
the raw value of this TTLType.
Since:
8.6, XC10 2.5

---

toString

`public String **toString**()`

Returns a string representation of the `TTLType`.

Overrides:
`toString` in class `Object`

Returns:
a string representation of the `TTLType`.

---

com.ibm.websphere.objectgrid

Interface StateManager

public interface **StateManager**

The StateManager can be used to retrieve the availability state of an ObjectGrid. Use the `StateManagerFactory.getStateManager()` method to retrieve a StateManager instance.

Since:

WAS XD 6.1.0.3, XC10

Method Summary

AvailabilityState	**getObjectGridState**(ObjectGrid objectGrid) Get the AvailabilityState of an ObjectGrid.
void	**setObjectGridState**(AvailabilityState state, ObjectGrid objectGrid) Set the AvailabilityState for an ObjectGrid.

Method Detail

getObjectGridState

AvailabilityState **getObjectGridState**(ObjectGrid objectGrid)

Get the AvailabilityState of an ObjectGrid. A random shard within the ObjectGrid is chosen for reporting availability state.

Parameters:

`objectGrid` - the availability state of the specified remote ObjectGrid will be retrieved

Returns:

the AvailabilityState of the remote ObjectGrid

Throws:

`IllegalArgumentException.` - If parameter objectGrid, is either null or it is of type 'LOCAL'. See `ObjectGrid.getObjectGridType()`.

TargetNotAvailableException - if there are no active shards for the ObjectGrid specified.

---

setObjectGridState

```
void setObjectGridState(AvailabilityState state,
                        ObjectGrid objectGrid)
```

Set the AvailabilityState for an ObjectGrid. Each shard in the ObjectGrid will be transitioned to the state specified. This method does not return until each shard in the ObjectGrid has transitioned to the AvailabilityState specified or if it times-out.

Parameters:

`state` - the AvailabilityState to transition to.
`objectGrid` - the ObjectGrid to transaction to the specified AvailabilityState.

Throws:

`IllegalArgumentException.` -

1. If parameter ObjectGrid. is either null or is of type 'LOCAL'. See `ObjectGrid.getObjectGridType()`.
2. If parameter AvailabilityState is null.

---

com.ibm.websphere.objectgrid

Class SessionNotReentrantException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.TransactionException
          com.ibm.websphere.objectgrid.SessionNotReentrantException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class SessionNotReentrantException
extends TransactionException
```

A Session object can only be used by a single thread concurrently to perform map operations. If a thread tries to execute a map operation (for example, call a method on ObjectMap interface) while another thread is already executing a map operation for the Session, then this exception is thrown.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Field Summary

Fields inherited from class com.ibm.websphere.objectgrid.TransactionException
ivTransactionRolledBack

Constructor Summary

SessionNotReentrantException(String message, boolean rolledBack) Constructs a new SessionNotReentrantException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.TransactionException
isTransactionActive, wasTransactionRolledBack

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

SessionNotReentrantException

```
public SessionNotReentrantException(String message,
                                    boolean rolledBack)
```

Constructs a new SessionNotReentrantException with the specified detail message and a special indication of whether the transaction was rolled back as a result of this exception. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`rolledBack` - A value of `true` indicates the transaction was rolled back.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage(), TransactionException.wasTransactionRolledBack()

com.ibm.websphere.objectgrid

Interface Session

public interface **Session**

This interface represents a session container for `ObjectMaps`. A thread must get its own Session object to interact with ObjectGrid. You can think of this interface as a session that can only be used by a single thread at a time. A Session itself is shareable across threads so long as only one thread uses it at a time. However, if a J2EE connection/transaction infrastructure is being used, that won't be shareable across threads and will prevent the Session object from being shared across threads. A good analogy for this object is a JDBC connection to a database. For best performance, use the `close()` method to close the session once it is no longer required.

Since:

WAS XD 6.0, XC10

See Also:

`ObjectGrid.getSession()`, `ObjectGrid.getSession(Subject)`, `ObjectGrid.getSession(CredentialGenerator)`

Nested Class Summary

static class	**Session.TxCommitProtocol** The commit protocols that can be used to commit the Session's transaction

Field Summary

static long	**DEFAULT_RETRY_TIMEOUT**
static int	**TRANSACTION_NO_TIMEOUT** A special value for the timeout parameter of the `setTransactionTimeout(int)` method.

static int	**TRANSACTION_READ_COMMITTED** A transaction isolation level constant indicating that dirty reads are prevented; non-repeatable reads and phantom reads can occur.
static int	**TRANSACTION_READ_UNCOMMITTED** A transaction isolation level constant indicating that dirty reads, non-repeatable reads and phantom reads can occur.
static int	**TRANSACTION_REPEATABLE_READ** A transaction isolation level constant indicating that dirty reads and non-repeatable reads are prevented; phantom reads can occur.
static String	**TRANSACTION_TYPE_DEFAULT** A string indicating the default transaction type

Method Summary

void	**begin**() Begins a new transaction.
void	**beginNoWriteThrough**() Starts a new transaction that does not write changes through to a `Loader` or ObjectGrid server.
void	**close**() Closes this session, freeing all resources that are held.
void	**commit**() Commits a transaction.
com.ib	

m.websphere.objectgrid.query.ObjectQuery	**createObjectQuery**(String qlString) Creates an instance of an object query for executing a query over the ObjectMaps visible to this session.
void	**flush**() Forces the current changes in the Session to the Loader or ObjectGrid server.
com.ibm.websphere.objectgrid.em	**getEntityManager**() Retrieve the EntityManager associated with this Session.

.EntityManager	
ObjectMap	**getMap**(String cacheName) Returns the ObjectMap for the specified name.
ObjectGrid	**getObjectGrid**() Returns the ObjectGrid that owns this session.
long	**getRequestRetryTimeout**() Retrieves the current request retry timeout for this session.
com.ibm.websphere.objectgrid.Sessi	**getSessionHandle**() Retrieves a handle for this session.

onHandle	
int	**getTransactionIsolation**() Retrieves the current transaction isolation level for this session.
int	**getTransactionTimeout**() Gets current transaction timeout for this Session.
String	**getTransactionType**() Retrieves the transaction type that is set with the setTransactionType(String) method.
Session.TxCommitProtocol	**getTxCommitProtocol**() Retrieve the current commit protocol for this Session.
TxID	**getTxID**() Gets the `TxID` transaction identifier, if a transaction is active.
boolean	**isCommitting**() Returns whether the current session transaction is performing a session `commit()` operation.
boolean	**isFlushing**() Returns whether the current session transaction is performing a session `flush()` operation.
boo	**isMarkedRollbackOnly**() Returns whether or not the current active session transaction is marked as being

boolean	rollback only as a result of a prior call to the `markRollbackOnly(Throwable)` method on this Session.
boolean	**`isSessionHandleSet`**`()` Determines if a SessionHandle is currently set on this Session.
boolean	**`isTransactionActive`**`()` Determines if a transaction is currently active.
boolean	**`isWriteThroughEnabled`**`()` Returns whether the current session transaction is writing through to the back end `Loader` or ObjectGrid server(`true`), or if the changes are only applying to the `BackingMap` (`false`) or client respectively.
void	**`markRollbackOnly`**`(Throwable t)` Marks the current transaction as being rollback only.
void	**`processLogSequence`**`(LogSequence logSequence)` Processes a `LogSequence`.
void	**`rollback`**`()` Rolls back a transaction.
void	**`setRequestRetryTimeout`**`(long requestRetryTimeout)` Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions.
void	**`setSessionHandle`**`(com.ibm.websphere.objectgrid.SessionHandle target)` Apply a SessionHandle to this session.
void	**`setTransactionIsolation`**`(int level)` Attempts to change the transaction isolation level for this session.
void	**`setTransactionTimeout`**`(int timeout)` Sets the transaction timeout for the next transaction started by this Session object to a specified number of seconds.
void	**`setTransactionType`**`(String tranType)` Sets the transaction type for future transactions.
vo	**`setTxCommitProtocol`**`(Session.TxCommitProtocol protocol)`

id	Set the commit protocol to be used when committing this Session's transaction.
boolean	**transactionTimedOut**() Determines whether the current session transaction has timed out.

Field Detail

TRANSACTION_TYPE_DEFAULT

static final String **TRANSACTION_TYPE_DEFAULT**

A string indicating the default transaction type

See Also:
Constant Field Values

TRANSACTION_NO_TIMEOUT

static final int **TRANSACTION_NO_TIMEOUT**

A special value for the timeout parameter of the `setTransactionTimeout(int)` method. This special value is used to indicate that the next transaction started by this Session is allowed unlimited amount of time.

See Also:
`setTransactionTimeout(int)`, Constant Field Values

DEFAULT_RETRY_TIMEOUT

static final long **DEFAULT_RETRY_TIMEOUT**

See Also:
Constant Field Values

TRANSACTION_REPEATABLE_READ

static final int **TRANSACTION_REPEATABLE_READ**

A transaction isolation level constant indicating that dirty reads and non-repeatable reads are prevented; phantom reads can occur. This level prohibits a transaction from reading an uncommitted cache entry, and it also prohibits the situation where one transaction reads an entry, a second transaction alters the entry, and the first transaction rereads the entry, getting different values the second time (a "non-repeatable read").

Since:
WAS XD 6.1.0.1
See Also:
`setTransactionIsolation(int)`, Constant Field Values

TRANSACTION_READ_COMMITTED

```
static final int TRANSACTION_READ_COMMITTED
```

A transaction isolation level constant indicating that dirty reads are prevented; non-repeatable reads and phantom reads can occur. This level only prohibits a transaction from reading a cache entry with uncommitted changes in it.

Since:
WAS XD 6.1.0.1
See Also:
`setTransactionIsolation(int)`, Constant Field Values

TRANSACTION_READ_UNCOMMITTED

```
static final int TRANSACTION_READ_UNCOMMITTED
```

A transaction isolation level constant indicating that dirty reads, non-repeatable reads and phantom reads can occur. This level allows a cache entry changed by one transaction to be read by another transaction before any changes in that entry have been committed (a "dirty read"). If any of the changes are rolled back, the second transaction will have retrieved an invalid entry.

Since:
WAS XD 6.1.0.1
See Also:
`setTransactionIsolation(int)`, Constant Field Values

Method Detail

beginNoWriteThrough

```
void beginNoWriteThrough()
                      throws TransactionAlreadyActiveException,
                             TransactionException
```

Starts a new transaction that does not write changes through to a `Loader` or ObjectGrid server.

Changes made by the session transaction started by this method are only applied to the `BackingMap` and not given to the `Loader`. This method can be used to apply changes made in a peer cache to the local `BackingMap` only. In addition, with a distributed map, this method can be used to start a session transaction which changes will only be applied to the client `BackingMap`, but not the `BackingMap` on the server side.

Throws:
`TransactionAlreadyActiveException` - if there is already an active transaction
`TransactionException` - a `TransactionCallbackException` occurred or some other error occurred starting a new transaction

---

getMap

```
ObjectMap getMap(String cacheName)
                 throws UndefinedMapException
```

Returns the ObjectMap for the specified name.

The ObjectMap is used to retrieve and modify values in the BackingMap. Multiple invocations of this method on the same Session object will always return the same object.

This method can also be used to create a BackingMap and its associated ObjectGrid after ObjectGrid initialization. If cacheName does not match the name of a previously created

map, a name comparison will be executed against template maps that have been configured. The ObjectMap and BackingMap will be created if the name matches the regular expression of a template.

Required Client Permission: `ObjectGridPermission.DYNAMIC_MAP` (when creating a new map from a template)

Parameters:

`cacheName` - name of desired map

Returns:

ObjectMap the transactional interface to modify values in the map

Throws:

`UndefinedMapException` - if the map is not defined.

See Also:

`ObjectGrid.defineMap(String)`, `ObjectMap`

---

begin

```
void begin()
          throws TransactionAlreadyActiveException,
                 TransactionException
```

Begins a new transaction.

Throws:

`TransactionAlreadyActiveException` - if this method is invoked with an active transaction

`TransactionException` - a `TransactionCallbackException` occurred or some other error occurred starting a new transaction

---

commit

```
void commit()
          throws NoActiveTransactionException,
                 TransactionException
```

Commits a transaction.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

`TransactionException` - if an error occurred during commit processing, see the caused by to determine the root error

See Also:

`markRollbackOnly(Throwable)`

---

rollback

```
void rollback()
          throws NoActiveTransactionException,
                 TransactionException
```

Rolls back a transaction.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

`TransactionException` - if an error occurred during rollback processing, see the caused by to determine the root error

---

flush

```
void flush()
        throws NoActiveTransactionException,
               TransactionException
```

Forces the current changes in the Session to the `Loader` or ObjectGrid server. This method does not commit the changes, it just applies the changes.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction
`TransactionException` - if an error occurred during flush processing, see the caused by to determine the root error

---

getObjectGrid

```
ObjectGrid getObjectGrid()
```

Returns the ObjectGrid that owns this session.

Returns:

the owning ObjectGrid instance.

---

isTransactionActive

```
boolean isTransactionActive()
```

Determines if a transaction is currently active.

Returns:

true if a transaction is currently active for this session.

Since:

WAS XD 6.1 FIX3

---

getTxID

```
TxID getTxID()
        throws NoActiveTransactionException
```

Gets the `TxID` transaction identifier, if a transaction is active.

Returns:

The current TxID object.

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

---

isWriteThroughEnabled

```
boolean isWriteThroughEnabled()
```

Returns whether the current session transaction is writing through to the back end `Loader` or ObjectGrid server(`true`), or if the changes are only applying to the `BackingMap` (`false`) or client respectively.

Returns:

`true`, if write through is enabled

See Also:

`begin()`, `beginNoWriteThrough()`

---

setTransactionType

```
void setTransactionType(String tranType)
```

Sets the transaction type for future transactions.

After this method is called, all future transactions will have the same type until another transaction type is set. If no transaction type is set, the default transaction type `TRANSACTION_TYPE_DEFAULT` will be used.

Transaction types are used mainly for statistical data tracking purpose. Users can predefine types of transactions that will be executed in an application. The idea is to categorize transactions with the same characteristics to one category (type), so one transaction response time statistics can be used to track each transaction type. This approach is useful when your application has different types of transactions. Some types of transactions, such as update transactions, process longer than others transactions, such as read-only transactions. By using the transaction type, different transactions are tracked by different statistics, so the statistics can be more useful.

Parameters:

`tranType` - the transaction type for future transactions.

See Also:

`TRANSACTION_TYPE_DEFAULT`

---

getTransactionType

```
String getTransactionType()
```

Retrieves the transaction type that is set with the setTransactionType(String) method.

Returns:

the transaction type for the session.

Since:

7.1.1.1

---

processLogSequence

```
void processLogSequence(LogSequence logSequence)
                        throws NoActiveTransactionException,
                               UndefinedMapException,
                               ObjectGridException
```

Processes a `LogSequence`.

Each `LogElement` within the `LogSequence` will be examined and the appropriate operation (insert, update, invalidate, etc) will be performed against the `BackingMap` identified by the `LogSequence`'s map name. An ObjectGrid Session must be active before this method is invoked. The caller is responsible for issuing the appropriate commit or rollback invocation to complete the Session. Autocommit processing is not available for this method invocation.

The main use of this method is for processing a `LogSequence` that was received by a remote JVM. For example, using the Distributed Commit support, the `LogSequences` associated with a given committed Session are distributed to other listening `ObjectGrids` in other JVMs. After receiving the `LogSequences` at the remote JVM, the listener could start a Session using `beginNoWriteThrough()`, invoke this method, and commit the Session transaction.

Parameters:

`logSequence` - LogSequence of changes to be applied to an active transaction

Throws:

`NoActiveTransactionException` - if this method is invoked with no active transaction

`UndefinedMapException` - if the map referenced by the LogSequence cannot be found

ObjectGridException - if the LogSequence elements cannot be processed

See Also:

beginNoWriteThrough(), LogSequence, ObjectGridEventListener

---

isFlushing

`boolean` **`isFlushing`**`()`

Returns whether the current session transaction is performing a session `flush()` operation. It is helpful to know if a session `flush()` is active (`true`), or if only an `ObjectMap.flush()` is in progress (returns `false` in this case).

Returns:

true, if the session is executing a session `flush()` call.

Since:

WAS XD 6.0.1

See Also:

flush(), ObjectMap.flush()

---

isCommitting

`boolean` **`isCommitting`**`()`

Returns whether the current session transaction is performing a session `commit()` operation. It is helpful to know if a session commit is active (`true`), or if an `ObjectMap.flush()` or session `flush()` is in progress (returns `false` in these cases).

Returns:

true, if session is executing a session `commit()` call.

Since:

WAS XD 6.0.1

See Also:

commit(), flush(), ObjectMap.flush()

---

markRollbackOnly

`void` **`markRollbackOnly`**`(Throwable t)`
`throws NoActiveTransactionException`

Marks the current transaction as being rollback only.

Marking a transaction rollback only ensures that even if the `commit()` method is called for this session transaction, the transaction is rolled back. A rollback is typically done when either ObjectGrid itself or the application knows that data corruption could occur if the `commit()` method was allowed to commit the transaction. Once this method is called, the Throwable object that is passed to it is chained to the `TransactionException` that is thrown if the commit method is ever called. Any subsequent calls to this method for the current active transaction is ignored (e.g. only the first call that passes a non null Throwable reference is used). Once the transaction is completed, the rollback only mark is removed so that the next transaction that is started using this session can be committed.

Parameters:

`t` - the Throwable that caused this method to be called.

Throws:

NoActiveTransactionException - if there is no active transaction for this Session.

Since:

WAS XD 6.0.1

See Also:

commit(), TransactionException

isMarkedRollbackOnly

`boolean` **`isMarkedRollbackOnly`**`()`

Returns whether or not the current active session transaction is marked as being rollback only as a result of a prior call to the `markRollbackOnly(Throwable)` method on this Session.

Returns:
true if and only if current session transaction is marked rollback only.
Since:
WAS XD 6.0.1
See Also:
`markRollbackOnly(Throwable)`

---

setTransactionTimeout

`void` **`setTransactionTimeout`**`(int timeout)`

Sets the transaction timeout for the next transaction started by this Session object to a specified number of seconds.

This method does not affect the transaction timeout of any transactions previously started by this Session. It only affects transactions that are started after this method is called. If this method is never called, the ObjectGrid configured transaction timeout value is used.

Parameters:
`timeout` - is the transaction timeout value in seconds. Use the special value `TRANSACTION_NO_TIMEOUT` if transaction is allowed unlimited amount of time and no transaction timeout should occur.
Since:
WAS XD 6.0.1
See Also:
`TRANSACTION_NO_TIMEOUT`, `ObjectGrid.setTxTimeout(int)`, `TransactionTimeoutException`

---

getTransactionTimeout

`int` **`getTransactionTimeout`**`()`

Gets current transaction timeout for this Session.

The transaction timeout value returned is the value that was configured for the `ObjectGrid` using `ObjectGrid.setTxTimeout(int)` or the value passed to `setTransactionTimeout(int)` to override the value configured on `ObjectGrid`. The return value is in seconds.

Returns:
timeout value in seconds.
Since:
WAS XD 6.0.1
See Also:
`setTransactionTimeout(int)`, `ObjectGrid.setTxTimeout(int)`

---

getTransactionIsolation

`int` **`getTransactionIsolation`**`()`

Retrieves the current transaction isolation level for this session.

Returns:
one of the following `Session` constants: `TRANSACTION_READ_UNCOMMITTED`, `TRANSACTION_READ_COMMITTED` or `TRANSACTION_REPEATABLE_READ`

Since:
WAS XD 6.1.0.1

---

transactionTimedOut

`boolean` **`transactionTimedOut`**`()`

Determines whether the current session transaction has timed out.

Returns:
true if and only if transaction has timed out.

Since:
WAS XD 6.0.1

See Also:
`setTransactionTimeout(int)`

---

createObjectQuery

`com.ibm.websphere.objectgrid.query.ObjectQuery` **`createObjectQuery`**`(String qlString) throws com.ibm.websphere.objectgrid.query.ObjectQueryException`

Creates an instance of an object query for executing a query over the ObjectMaps visible to this session.

When ObjectGrid security is enabled, this method requires an com.ibm.websphere.objectgrid.security.ObjectGridPermission with action "query".

Required Client Permission: `ObjectGridPermission.QUERY`

Parameters:
`qlString` - a query string

Returns:
the new query instance.

Throws:
`com.ibm.websphere.objectgrid.query.ObjectQueryException` - if an error occurs creating the object query.

Since:
WAS XD 6.1

---

getEntityManager

`com.ibm.websphere.objectgrid.em.EntityManager` **`getEntityManager`**`()`

Retrieve the EntityManager associated with this Session. Each session is associated with a single EntityManager instance. Repeated calls to this method on the same Session instance will result in the same EntityManager instance.

Returns:
this session's EntityManager instance.

Since:
WAS XD 6.1

---

setTransactionIsolation

```
void setTransactionIsolation(int level)
```

Attempts to change the transaction isolation level for this session. The constants defined in the `Session` interface are the possible transaction isolation levels.

This method should normally be invoked prior to beginning a transaction. Invoking after a transaction has started may result in an exception.

Parameters:

`level` - one of the following `Session` constants: `TRANSACTION_READ_UNCOMMITTED`, `TRANSACTION_READ_COMMITTED` or `TRANSACTION_REPEATABLE_READ`

Since:

WAS XD 6.1.0.1

---

getSessionHandle

```
com.ibm.websphere.objectgrid.SessionHandle getSessionHandle()
```

Retrieves a handle for this session.

A SessionHandle contains partition information for the current session and can be re-applied to a new session using the `setSessionHandle(SessionHandle)` method. A SessionHandle is only applicable for ObjectGrids using per-container partition placement. If `setSessionHandle(SessionHandle)` is not called before invoking this method, a Session Handle is selected using the properties configured in the `ClientProperties`. If there are no per-container partition placement mapsets or more than one in the ObjectGrid, an `IllegalStateException` is thrown.

Returns:

the SessionHandle for this session

Throws:

`IllegalStateException` - if this method is called in an invalid environment.

Since:

WAS XD 6.1.0.3

---

setSessionHandle

```
void setSessionHandle(com.ibm.websphere.objectgrid.SessionHandle target)
                      throws TargetNotAvailableException
```

Apply a SessionHandle to this session.

Parameters:

`target` - the SessionHandle to apply or null to disassociate a SessionHandle from this session.

Throws:

`TargetNotAvailableException` - when the target is no longer available.

`IllegalStateException` - if the Session has modified some maps already and the SessionHandle has already been set or if this method is called in an invalidate environment.

Since:

WAS XD 6.1.0.3

---

setRequestRetryTimeout

```
void setRequestRetryTimeout(long requestRetryTimeout)
```

Set the request retry timeout to indicate how long to retry a request (in milliseconds) when recoverable failures occur, such as fail-over exceptions. A request will timeout

when either the request timeout expires or the transaction timeout expires, whichever expires first.

A value of 0 indicates that all requests should fail immediately and avoid any retry logic. Exceptions that cannot succeed even if tried again such as DuplicateKeyException exceptions will be thrown immediately.

A value of -1 indicates that the request retry timeout is not set, meaning that the request duration is governed by the request retry timeout set on the ClientProperties. If the ClientProperties is also set to -1, then the request retry timeout is governed by the transaction timeout.

Parameters:

`requestRetryTimeout` - the duration in milliseconds retry a client request, 0 if the request should fail immediately or -1 if the request timeout is not set.

Since:

7.0

See Also:

`ClientProperties.setRequestRetryTimeout(long)`, `setTransactionTimeout(int)`

---

getRequestRetryTimeout

`long` **`getRequestRetryTimeout`**`()`

Retrieves the current request retry timeout for this session. Returns -1 if it was not set.

Returns:

the duration in milliseconds retry a client request, 0 if the request should fail immediately or -1 if the request timeout is not set.

Since:

7.0

---

isSessionHandleSet

`boolean` **`isSessionHandleSet`**`()`

Determines if a SessionHandle is currently set on this Session.

Returns:

true if a SessionHandle is currently set on this session.

Since:

7.1

---

close

`void` **`close`**`()`

Closes this session, freeing all resources that are held. Once closed, this session must be discarded. Use one of the `ObjectGrid.getSession()` methods to retrieve a new session. If the session has an active transaction, the transaction will be rolled back and the session resources are not freed.

Throws:

`ObjectGridRuntimeException` - thrown if there is a problem releasing resources held by this session.

Since:

7.1.1

---

setTxCommitProtocol

`void` **`setTxCommitProtocol`**`(`[`Session.TxCommitProtocol`](#) `protocol)`

Set the commit protocol to be used when committing this Session's transaction. The constants defined in the `TxCommitProtocol` enum are the possible commit protocols.

This method should normally be invoked prior to beginning a transaction. Invoking after a transaction has started will result in an exception.

Parameters:

`protocol` - one of the following constants `TxCommitProtocol.ONEPHASE` or `TxCommitProtocol.TWOPHASE`

Since:

8.6, XC10 2.5

---

getTxCommitProtocol

[`Session.TxCommitProtocol`](#) **`getTxCommitProtocol`**`()`

Retrieve the current commit protocol for this Session.

Returns:

one of the following constants `TxCommitProtocol.ONEPHASE` or `TxCommitProtocol.TWOPHASE`

Since:

8.6, XC10 2.5

---

com.ibm.websphere.objectgrid

Enum Session.TxCommitProtocol

java.lang.Object
  java.lang.Enum<Session.TxCommitProtocol>
    com.ibm.websphere.objectgrid.Session.TxCommitProtocol

All Implemented Interfaces:

Serializable, Comparable<Session.TxCommitProtocol>

Enclosing interface:

Session

```
public static enum Session.TxCommitProtocol
extends Enum<Session.TxCommitProtocol>
```

The commit protocols that can be used to commit the Session's transaction

Since:

8.6, XC10 2.5

Enum Constant Summary

ONEPHASE A commit protocol constant indicating that the Session transaction can read from multiple partitions but can only write to a single partition.
TWOPHASE A commit protocol constant indicating that the Session transaction can read and write from multiple partitions.

Method Summary

static Session.TxCommitProtocol	**valueOf**(String name) Returns the enum constant of this type with the specified name.

Protocol	
staticSession.TxCommitProtocol[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

ONEPHASE

public static final Session.TxCommitProtocol **ONEPHASE**

A commit protocol constant indicating that the Session transaction can read from multiple partitions but can only write to a single partition. A TransactionException is thrown if the transaction writes to multiple partitions. The transaction is committed using the one-phase commit protocol.

TWOPHASE

public static final Session.TxCommitProtocol **TWOPHASE**

A commit protocol constant indicating that the Session transaction can read and write from multiple partitions. The transaction is committed using the two-phase commit protocol. If the transaction only writes to a single partition then the transaction is

committed using the one-phase commit protocol.

Method Detail

values

```
public static Session.TxCommitProtocol[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (Session.TxCommitProtocol c : Session.TxCommitProtocol.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

---

valueOf

```
public static Session.TxCommitProtocol valueOf(String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
`name` - the name of the enum constant to be returned.
Returns:
the enum constant with the specified name
Throws:
`IllegalArgumentException` - if this enum type has no constant with the specified name
`NullPointerException` - if the argument is null

---

com.ibm.websphere.objectgrid

Class ReplicationVotedToRollbackTransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.TransactionCallbackException
          com.ibm.websphere.objectgrid.ReplicationVotedToRollbackTransactionException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **ReplicationVotedToRollbackTransactionException**
extends TransactionCallbackException

This exception is thrown when a transaction was rolled back because some/all of the replicas failed to apply the transaction when in synchronous replication mode.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

ReplicationVotedToRollbackTransactionException() Constructs a new ReplicationVotedToRollbackTransactionException with `null` as its detail message.
ReplicationVotedToRollbackTransactionException(String message) Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message.
ReplicationVotedToRollbackTransactionException(String message, Throwable cause) Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message and cause.
ReplicationVotedToRollbackTransactionException(Throwable cause) Constructs a new ReplicationVotedToRollbackTransactionException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException()
```

Constructs a new ReplicationVotedToRollbackTransactionException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException(String message)
```

Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
ObjectGridException.initCause(Throwable), Throwable.getMessage()

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException(String message,
                                                      Throwable cause)
```

Constructs a new ReplicationVotedToRollbackTransactionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ReplicationVotedToRollbackTransactionException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
ObjectGridException.getCause(), Throwable.getMessage()

ReplicationVotedToRollbackTransactionException

```
public ReplicationVotedToRollbackTransactionException(Throwable cause)
```

Constructs a new ReplicationVotedToRollbackTransactionException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ReplicationVotedToRollbackTransactionExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Class ReadOnlyException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.ReadOnlyException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class ReadOnlyException
extends ObjectGridException
```

This exception is thrown when an attempt is made to modifying operations on a read only maps.

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Constructor Summary

ReadOnlyException() Constructs a new ReadOnlyException with `null` as its detail message.
ReadOnlyException(String message) Constructs a new ReadOnlyException with the specified detail message.
ReadOnlyException(String message, Throwable cause) Constructs a new ReadOnlyException with the specified detail message and cause.
ReadOnlyException(Throwable cause) Constructs a new ReadOnlyException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ReadOnlyException

```
public ReadOnlyException()
```

Constructs a new ReadOnlyException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
`ObjectGridException.initCause(Throwable)`

ReadOnlyException

```
public ReadOnlyException(String message)
```

Constructs a new ReadOnlyException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

ReadOnlyException

```
public ReadOnlyException(String message,
                         Throwable cause)
```

Constructs a new ReadOnlyException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ReadOnlyException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
`ObjectGridException.getCause()`, `Throwable.getMessage()`

ReadOnlyException

```
public ReadOnlyException(Throwable cause)
```

Constructs a new ReadOnlyException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ReadOnlyExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Interface PartitionManager

public interface **PartitionManager**

This interface will be used for calculating the proper partition for a given input key. The set of partitions is determined by the BackingMap configuration.

Since:
WAS XD 6.0.1, XC10

See Also:
BackingMap.getPartitionManager()

Method Summary

int	**getNumOfPartitions**() Returns the number of configured partitions for this PartitionManager.
int	**getPartition**(Object key) Obtains a 0-based partition number determined by the input parameter's hashCode() method.
List	**getPartitionLists**(List keyList) This method is very similar to `getPartitions(List)`, except it returns the keys organized by the partition identifiers.
List	**getPartitions**(List keyList) Obtains the 0-based partition numbers for each of the keys in the input `List` of keys.
List	**partitionLogSequence**(LogSequence ls) Partitions a `LogSequence` based on the partitioning algorithm for the Map.

Method Detail

getPartition

int **getPartition**(Object key)

Obtains a 0-based partition number determined by the input parameter's hashCode() method.

Parameters:
`key` - Individual key used to determine partition (can not be null)

Returns:
int 0-based partition number

getPartitions

List **getPartitions**(List keyList)

Obtains the 0-based partition numbers for each of the keys in the input `List` of keys. Each object in the returned list of partition identifiers is an instance of `java.lang.Integer`.

Parameters:
`keyList` - Ordered list of keys
Returns:
List of partition identifiers that corresponds to the input list of keys

---

getPartitionLists

List **getPartitionLists**(List keyList)

This method is very similar to `getPartitions(List)`, except it returns the keys organized by the partition identifiers. The return value is a `List` of `Lists`. The outer `List` is an ordered `List` of the partition numbers, with the first entry in the `List` corresponding to partition 0. The inner `Lists` contain the keys from the input parameter that correspond to that partition identifier.

The return value will always contain a `List` object. Either the outer or the inner `Lists` may contain zero elements, but the `List` objects themselves will not be `null`.

Parameters:
`keyList` - Ordered list of keys
Returns:
List of Lists that correspond to the 0-based partition numbers, with each inner List containing the set of keys that parse to that partition number.

---

partitionLogSequence

List **partitionLogSequence**(LogSequence ls)

Partitions a `LogSequence` based on the partitioning algorithm for the Map.

Parameters:
`ls` - LogSequence that needs to be partitioned
Returns:
List of partitioned LogSequences. The first LogSequence in the List corresponds to the first partition, etc.

---

getNumOfPartitions

int **getNumOfPartitions**()

Returns the number of configured partitions for this PartitionManager.

Returns:
the number of configured partitions

---

com.ibm.websphere.objectgrid

Interface ObjectMap

public interface **ObjectMap**

This is a handle to a named Map. Maps should have homogeneous keys and values. An instance of this ObjectMap can only be used by the thread that is currently associated with the Session that was used to get this ObjectMap instance. Both Session and ObjectMap objects are not allowed to be shared by multiple threads concurrently.

The `CopyMode` setting on the map determines whether or not a copy of the value is returned by get methods. It also determines whether or not a copy of the committed value is made at commit time. The `LockStrategy` setting for the map determines whether or not a lock is obtained for each map entry accessed by the transaction, the lock mode of the lock obtained, and when the lock is obtained.

Each data access method includes a "Specification details" table that includes the following information:

Required permission:	The permission required to use the API.
Pessimistic read lock acquired:	The type of lock that is acquired when using pessimistic locking with repeatable read transaction isolation.
Pessimistic read lock held:	The type of lock that is held for the duration of the transaction when using pessimistic locking with repeatable read transaction isolation. Locks can be upgraded but not demoted during a transaction.
Transaction:	The state of the transaction when invoking the API. • **Manual** - The transaction is explicitly demarcated using the associated `Session` and the specified API is used within the scope of that transaction. • **Automatic** - The transaction is automatically demarcated. This is also referred to as an auto-commit transaction or API which the associated `Session.isTransactionActive()` is false when invoking the API.
Cache tier:	Identifies the map cache tiers that are included when fetching or updating cache entries in the call and under what circumstances. The following tiers are available for client maps: • Transactional Cache • Near Cache (if enabled) • Server Cache • Loader (if enabled) The following tiers are available for local maps: • Transactional Cache • Local Cache • Loader (if enabled)

Since:

WAS XD 6.0, XC10

See Also:

Session.getMap(String), BackingMap.setCopyMode(CopyMode, Class), BackingMap.setLockStrategy(LockStrategy)

Nested Class Summary

static class	**ObjectMap.PutMode** Identifies the operation mode of the put(Object, Object), putAll(Map) , JavaMap.put(Object, Object) and JavaMap.putAll(Map) methods.

Field Summary

static long	**QUEUE_TIMEOUT_INFINITE** Used as a parameter on the getNextKey(long) method, specifies the method should block until a key becomes available.
static long	**QUEUE_TIMEOUT_NONE** Used as a parameter on the getNextKey(long) method, specifies to return a null value if the map is empty.
static int	**TTL_FOREVER** A constant indicating the time-to-live value is "forever".
static int	**USE_DEFAULT** A constant indicating the time-to-live value or lock timeout value is the default setting.

Method Summary

v	**clear**()

void	Clear all keys from the Map.
void	**clearCopyMode**() Resets the CopyMode back to the one in the BackingMap.
boolean	**containsKey**(Object key) Returns `true` if this map contains a mapping for the specified key.
void	**flush**() Pushes the current set of changes for the `ObjectMap` instance to the `Loader` without committing the changes.
Object	**get**(Object key) Retrieves the object from the cache at the given key.
com.ibm.websphere.objectgrid.datagrid.AgentMan	**getAgentManager**() Returns the Agent Manager that allows DataGrid operations to be performed on the objects within this Map.

a g e r	
List	**getAll**(List keyList) Gets a list of entries from the map.
List	**getAllForUpdate**(List keyList) Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries.
c o m . i b m . w e b s p h e r e . p r o j e c t o r . m d . E n t i t y M e t a d a t a	**getEntityMetadata**() Retrieve the metadata for the entity associated with this map.
Object	**getForUpdate**(Object key) Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry.
Object	**getIndex**(String name) Returns a reference to the named index that can be used with this Map.

Object	**getIndex**(String name, boolean forUpdate) Returns a reference to the named index that can be used with this Map.
Map	**getJavaMap**() Returns an implementation of `java.util.Map` that is backed by this `ObjectMap`.
com.ibm.websphere.objectgrid.OutputFormat	**getKeyOutputFormat**() Retrieves the data format for all data access APIs that return cache keys for the current session.
int	**getMapType**() Returns the type of the underlying BackingMap.
String	**getName**() Returns the name of the ObjectMap as defined by the configuration.
Object	**getNextKey**(long timeout) Retrieves a key off the map in first-in-first-out (FIFO) insert order.
Ob	

int	**getPutMode**() Retrieve the current PutMode set for this ObjectMap instance.
com.ibm.websphere.objectgrid.OutputFormat	**getValueOutputFormat**() Retrieves the data format for all data access APIs that return cache values for the current session.
void	**insert**(Object key, Object value) Performs an explicit insert of a given entry.
void	**invalidate**(Object key, boolean isGlobal) Invalidates an entry in the cache based on the key parameter.
void	**invalidateAll**(Collection keyList, boolean isGlobal) Invalidate a set of cache entries based on the Collection of keys provided.
bo	

boolean	**lock**(Object key, com.ibm.websphere.objectgrid.LockMode lockMode) Obtains a lock for the given key.
List<Boolean>	**lockAll**(List keyList, com.ibm.websphere.objectgrid.LockMode lockMode) Obtains locks for the given keys.
Object	**put**(Object key, Object value) Puts the Object value into the cache at location represented by key.
void	**putAll**(Map map) Puts each of the Object value into the cache at location represented by key contained in the Map.
Object	**remove**(Object key) Removes the Object value from the cache represented by key.
void	**removeAll**(Collection keyList) Batch remove from the Map.
void	**setCopyMode**(CopyMode copyMode, Class valueInterface) Allows the CopyMode for the Map to be overridden on this map on this session only.
void	**setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data format for all data access APIs that return cache keys for the current session.
void	**setLockTimeout**(int seconds) Overrides the BackingMap's lock timeout for this ObjectMap.
void	**setPutMode**(ObjectMap.PutMode putMode) Allows the default operation for the put(Object, Object) and putAll(Map) methods to be changed, allowing the put to use the optimized upsert(Object, Object) and upsertAll(LinkedHashMap) implementations.
int	**setTimeToLive**(int ttl) Establishes the number of seconds that any given cache entry can live for, which is referred to as "time to live" or TTL.
void	**setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data format for all data access APIs that return cache values for the current session.

void	**touch**(Object key) Updates the last access time in the BackingMap without retrieving the value to the ObjectMap.
void	**update**(Object key, Object value) Performs an explicit update of a given entry.
void	**upsert**(Object key, Object value) Puts the Object value into the cache at location represented by key.
void	**upsertAll**(LinkedHashMap map) Puts each of the Object value into the cache at location represented by key contained in the Map.

Field Detail

TTL_FOREVER

```
static final int TTL_FOREVER
```

A constant indicating the time-to-live value is "forever".

See Also:
Constant Field Values

USE_DEFAULT

```
static final int USE_DEFAULT
```

A constant indicating the time-to-live value or lock timeout value is the default setting.

The default setting is to retain the time-to-live value for any existing map entry and to use the default value from BackingMap setting if a new map entry is being created.

For lock timeout override the default setting is to use the value defined on the BackingMap

See Also:
setLockTimeout(int), setTimeToLive(int), BackingMap.setTimeToLive(int), BackingMap.getTimeToLive(), BackingMap.setLockTimeout(int), Constant Field Values

---

QUEUE_TIMEOUT_NONE

```
static final long QUEUE_TIMEOUT_NONE
```

Used as a parameter on the getNextKey(long) method, specifies to return a null value if the map is empty.

See Also:
Constant Field Values

QUEUE_TIMEOUT_INFINITE

`static final long` **`QUEUE_TIMEOUT_INFINITE`**

Used as a parameter on the `getNextKey(long)` method, specifies the method should block until a key becomes available.

See Also:
Constant Field Values

Method Detail

getName

`String` **`getName`**`()`

Returns the name of the ObjectMap as defined by the configuration.

Returns:
name of ObjectMap

---

get

`Object` **`get`**`(Object key)`
`throws ObjectGridException`

Retrieves the object from the cache at the given key.

Whether or not a copy of the object is returned is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if a value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`
Pessimistic read lock held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found.

Parameters:
`key` - The entry to fetch
Returns:
the value, `null`, `SerializedValue` or `Tuple`
Throws:
`IllegalArgumentException` - if key is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.
See Also:
`containsKey(Object)`, `getForUpdate(Object)`, `CopyMode`

---

put

```
Object put(Object key,
           Object value)
           throws ObjectGridException
```

Puts the Object value into the cache at location represented by key.

The value will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the `setPutMode(PutMode)` property:

`ObjectMap.PutMode.INSERTUPDATE`	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
`ObjectMap.PutMode.UPSERT`	The value is put into the map using the specification of the `upsert(Object, Object)`.

Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.WRITE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - The entry to put into the map
`value` - The value to put into the map using the key

Returns:

If `ObjectMap.PutMode.INSERTUPDATE` is set, return the previous value in this transaction.
If `ObjectMap.PutMode.UPSERT` is set, the return value is null.

Throws:

`IllegalArgumentException` - if key is `null`, or if the map does not allow `null` values and value is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`CopyMode`

---

getForUpdate

```
Object getForUpdate(Object key)
                    throws ObjectGridException
```

Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry. See `LockStrategy.PESSIMISTIC` for additional information. Whether or not a copy of the object is returned is determined by the `CopyMode` setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if the value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.UPGRADABLE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

Parameters:

`key` - The entry to fetch

Returns:

the value, `null`, `SerializedValue` or `Tuple`

Throws:

`IllegalArgumentException` - if key is `null`

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`containsKey(Object)`, `get(Object)`, `CopyMode`, `LockStrategy.PESSIMISTIC`

---

remove

```
Object remove(Object key)
              throws ObjectGridException
```

Removes the Object value from the cache represented by key.

This removal will be pushed down to the BackingMap/Loader at commit time. If the key cannot be found in the map, a null value will be returned.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permissi	`MapPermission.REMOVE`

on:	
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - The entry to remove

Returns:

the current value at invocation time

Throws:

`IllegalArgumentException` - if key is `null`

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

getAll

```
List getAll(List keyList)
            throws ObjectGridException
```

Gets a list of entries from the map.

If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`
Pessimistic read lock held:	Yes
Transaction:	Automatic or manual
Cache tier:	For each key, progresses to all tiers until the key is found.

Parameters:

`keyList` - A list of keys for identifying which entries to fetch

Returns:

a list of values

Throws:

`IllegalArgumentException` - if keyList is null or contains a `null` element.

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`get(Object)`

---

getAllForUpdate

```
List getAllForUpdate(List keyList)
                     throws ObjectGridException
```

Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries. See `LockStrategy.PESSIMISTIC` for additional information. If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.UPGRADABLE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the keys are found and the appropriate locks are acquired.

Parameters:

`keyList` - A list of keys for identifying which entries to fetch

Returns:

a list of values

Throws:

`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`getAll(List)`, `getForUpdate(Object)`, `LockStrategy.PESSIMISTIC`

---

removeAll

```
void removeAll(Collection keyList)
               throws ObjectGridException
```

Batch remove from the Map. If a key in the list cannot be found, it will be ignored.

Specification details:

Required permission:	`MapPermission.REMOVE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`keyList` - A list of keys for identifying which entries to remove

Throws:

IllegalArgumentException - if keyList is null or contains a null element.
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

remove(Object)

---

putAll

```
void putAll(Map map)
            throws ObjectGridException
```

Puts each of the Object value into the cache at location represented by key contained in the Map.

The values will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the setPutMode(PutMode) property:

ObjectMap.PutMode.INSERTUPDATE	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
ObjectMap.PutMode.UPSERT	The values are put into the map using the specification of the upsertAll(LinkedHashMap).

Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See CopyMode for a description of each possible copy mode.

An existing Map object will be passed in to use for obtaining the keys and values to be inserted or updated into the existing Map.

Specification details:

Required permission:	MapPermission.WRITE
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use Session.beginNoWriteThrough() to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

map - The key/values to be put into the map.

Throws:

IllegalArgumentException - if map is null or contains a null key or if null values are not allowed and map contains a null value.
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

put(Object, Object)

---

invalidate

```
void invalidate(Object key,
                boolean isGlobal)
                throws ObjectGridException
```

Invalidates an entry in the cache based on the key parameter.

If the key's value has changes pending in the ObjectMap, it is the application's responsibility to flush these changes to the Loader before invalidation. If a flush is not performed prior to invoking the invalidate operation, all pending changes for this key will be removed from the ObjectMap. If the key cannot be found in the map, it will be ignored.

The isGlobal parameter is used to indicate which cache level is used to invalidate the entries. If isGlobal is true, when the transaction is committed, the key is removed from the BackingMap also. If a subsequent get operation is performed, the BackingMap will be skipped and the Loader will be used to get the data. If isGlobal is false, the entry is only invalidated in the ObjectMap (transactional cache). If a subsequent get operation is performed, the BackingMap can be used; and, if it's not in the BackingMap, the Loader will be used to get the data.

A typical use of isGlobal being false is when a large number of records are touched in a transaction and the application wants to evict records that are no longer used in the cache.

Specification details:

Required permission:	`MapPermission.INVALIDATE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit except the Loader. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps. Set the `isGlobal` parameter to `false` to limit the operation to the transaction cache tier.

Parameters:

`key` - Object representing the key to be used for cache entry invalidation
`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the ObjectMap (false).

Throws:

`IllegalArgumentException` - if key is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

invalidateAll

```
void invalidateAll(Collection keyList,
                   boolean isGlobal)
                   throws ObjectGridException
```

Invalidate a set of cache entries based on the Collection of keys provided. If a key in the collection cannot be found, it will be ignored.

Specification details:

Required permission:	`MapPermission.INVALIDATE`
Transaction:	Automatic or manual

actio
n:

Cach
e tier: Applied to all tiers during commit except the Loader. Use [Session.beginNoWriteThrough()](#) to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps. Set the `isGlobal` parameter to `false` to limit the operation to the transaction cache tier.

Parameters:

`keyList` - A Collection of keys representing the entries to be invalidated

`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the ObjectMap (false).

Throws:

[IllegalArgumentException](#) - if keyList is null or contains a `null` element.

[ObjectGridException](#) - if an error occurs during processing

[AccessControlException](#) - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

[invalidate(Object, boolean)](#)

---

setTimeToLive

`int` **`setTimeToLive`**`(int ttl)`

Establishes the number of seconds that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this `ObjectMap`, any previous value set by the `BackingMap.setTimeToLive(int)` method is overridden for this ObjectMap. If this method is never called on the ObjectMap, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from BackingMap setting if a new map entry is being created. If TTL is never set on the BackingMap, the cache entry can live "forever".

This method can only be used when the `TTLType` is set to `LAST_ACCESS_TIME` or `LAST_UPDATE_TIME` on the BackingMap. If this method is called on the ObjectMap and the `TTLType` is something other than `LAST_ACCESS_TIME` or `LAST_UPDATE_TIME`, an IllegalStateException is thrown.

Required permission: `MapPermission.INVALIDATE`

Parameters:

`ttl` - is the time-to-live value in seconds. The value must be >= -1. A value of 0 is used to indicate the cache entry can live "forever" and -1 to indicate to use default setting. Use of the constant `TTL_FOREVER` is recommended when "forever" is desired and the constant`USE_DEFAULT` is recommended when "use default" setting is desired.

Returns:

previous time-to-live value in seconds. The constant `TTL_FOREVER` and constant `USE_DEFAULT`can be used to determine if the previous TTL is one of the special values.

Throws:

[IllegalArgumentException](#) - if seconds argument is < -1.

[IllegalStateException](#) - if `BackingMap.getTtlEvictorType()` returns anything other than `TTLType.LAST_ACCESS_TIME` or `TTLType.LAST_UPDATE_TIME`.

[AccessControlException](#) - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

[TTL_FOREVER](#), [USE_DEFAULT](#), [BackingMap.setTimeToLive(int)](#), [TTLType.LAST_ACCESS_TIME](#), [TTLType.LAST_UPDATE_TIME](#)

---

update

```
void update(Object key,
            Object value)
            throws KeyNotFoundException,
                   ObjectGridException
```

Performs an explicit update of a given entry.

A `get` operation is not required prior to invoking the `update` method (unlike the `put` method). Also, an `update` invocation will never insert a new record. If a the map's `LockStrategy` is `LockStrategy.OPTIMISTIC` this method will implicitly get the entry so as to have the version value of the object for when this method was invoked. Whether or not a copy of the object is made when transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If a key cannot be found in the map during commit, a TransactionException will be thrown.

Specification details:

Required permission:	`MapPermission.WRITE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - Identifies the entry to be updated

`value` - The updated value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.

`KeyNotFoundException` - if the key cannot be found in the map

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`insert(Object, Object)`, `put(Object, Object)`, `CopyMode`, `LockStrategy.OPTIMISTIC`

---

insert

```
void insert(Object key,
            Object value)
            throws DuplicateKeyException,
                   ObjectGridException
```

Performs an explicit insert of a given entry.

The key must not exist before executing this method. Also, an `insert` invocation will never update an existing record. Whether or not a copy of the object is made when a transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If the key is already in the map, a TransactionException will be thrown during commit.

Specification details:

Required	`MapPermission.INSERT`

permissi on:	
Transact ion:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - Identifies the entry to be inserted

`value` - The value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.

`DuplicateKeyException` - if this entries already exists in the map

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

See Also:

`put(Object, Object)`, `update(Object, Object)`, `CopyMode`

---

getIndex

```
Object getIndex(String name)
                throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                       com.ibm.websphere.objectgrid.IndexNotReadyException
```

Returns a reference to the named index that can be used with this Map. This index cannot be shared between threads and works on the same rules as `Session`. The returned value should be cast to the right index interface such as `MapIndex`, `MapRangeIndex` or a custom index interface such as a geo spatial index.

Parameters:

`name` - The index name

Returns:

A reference to the index, it must be cast to the appropriate index interface.

Throws:

`IndexUndefinedException` - if the index is not defined on the BackingMap

`IndexNotReadyException` - if the index is a dynamic index and it is not ready

Since:

WAS XD 6.0.1

---

getIndex

```
Object getIndex(String name,
                boolean forUpdate)
                throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                       com.ibm.websphere.objectgrid.IndexNotReadyException
```

Returns a reference to the named index that can be used with this Map. This index cannot be shared between threads and works on the same rules as `Session`. The returned value should be cast to the right index interface such as `MapIndex`, `MapRangeIndex` or a custom index interface such as a geo spatial index.

Parameters:

`name` - The index name

`forUpdate` - if true, the returned index will always operate with forUpdate intent.

Returns:

A reference to the index, it must be cast to the appropriate index interface.

Throws:

`IndexUndefinedException` - if the index is not defined on the BackingMap

`IndexNotReadyException` - if the index is a dynamic index and it is not ready

Since:

WAS XD 6.1.0.1

---

flush

```
void flush()
        throws ObjectGridException
```

Pushes the current set of changes for the `ObjectMap` instance to the `Loader` without committing the changes. The changes are not propagated to the `BackingMap` either. This is useful for re-priming the `Loader`'s data without committing the current transaction and starting over.

Throws:

`ObjectGridException` - if an error occurs during processing

See Also:

`Session.flush()`

---

containsKey

```
boolean containsKey(Object key)
                    throws ObjectGridException
```

Returns `true` if this map contains a mapping for the specified key. ObjectGrid does not support null keys. If you configured the map to support `null` values, this method can be used to determine whether a key is contained in the map or not.

This API does not hold any locks when using pessimistic locking.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`
Pessimistic locks held:	None
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found.

Parameters:

`key` - key whose presence in this map is to be tested.

Returns:

`true` if this map contains a mapping for the specified key.

Throws:

`IllegalArgumentException` - if `null` key parameter is passed in

`ObjectGridException` - if an error occurs during processing

`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

getJavaMap

```
Map getJavaMap()
```

Returns an implementation of `java.util.Map` that is backed by this `ObjectMap`.

The returned `java.util.Map` implementation can be cast to `com.ibm.websphere.objectgrid.JavaMap` to be able to use the rest of the ObjectGrid

programming model, but with `java.util.Map`'s use of `RuntimeException`s instead of checked `ObjectGridException`s.

Returns:

a `java.util.Map` backed by this `ObjectMap`

See Also:

`Map`, `JavaMap`

---

touch

```
void touch(Object key)
           throws ObjectGridException
```

Updates the last access time in the BackingMap without retrieving the value to the ObjectMap.

The last access time is updated during commit. If the key does not exist in the BackingMap, a TransactionException will be returned during commit processing.
Specification details:

Required permission:	None
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - key to be touched

Throws:

`IllegalArgumentException` - if key is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

---

setCopyMode

```
void setCopyMode(CopyMode copyMode,
                 Class valueInterface)
                 throws TransactionAlreadyActiveException,
                        ObjectGridException
```

Allows the CopyMode for the Map to be overridden on this map on this session only.

This method allows an application to use an optimal CopyMode TRANSACTION by TRANSACTION as its needs dictate. The CopyMode cannot be changed during a transaction. There must be no active transaction when this method is called.

Parameters:

`copyMode` - must be one of the final static variables defined in `CopyMode`. See `CopyMode` class for an explanation of each mode and how the valueInterface is used for `CopyMode.COPY_ON_WRITE` .
`valueInterface` - the value interface Class object. Specify null in version 7.1 and later.

Throws:

`IllegalArgumentException` - if copyMode is `null` or COPY_ON_WRITE CopyMode is specified and the required value interface parameter is `null`
`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

ObjectGridException - if an error occurs during processing

See Also:

BackingMap.setCopyMode(CopyMode, Class), CopyMode

---

clearCopyMode

```
void clearCopyMode()
                throws TransactionAlreadyActiveException
```

Resets the CopyMode back to the one in the BackingMap.

This method is used to reverse a previous setCopyMode method call for this ObjectMap. This method can only be called when no transaction is active on the associated session.

Throws:

TransactionAlreadyActiveException - if a transaction is active and this map has already been used in the transaction.

See Also:

setCopyMode(CopyMode, Class)

---

getNextKey

```
Object getNextKey(long timeout)
                  throws ObjectGridException
```

Retrieves a key off the map in first-in-first-out (FIFO) insert order.

The entry is locked by the session such that other calls to getNextKey will not return the same key. The key can be used to remove or manipulate the value although leaving the entry will result in the key remaining at the beginning of the queue. This order is optimized for performance and is not guaranteed especially across partitions or in highly concurrent environments.

The return value is a `SerializedKey` when OutputFormat.RAW is set for the keys. The default key output format for maps that are associated with a KeySerializerPlugin is OutputFormat.RAW. The SerializedKey allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

Specification details:

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.EXCLUSIVE`
Pessimistic read lock held:	Yes
Transaction:	Automatic or manual

Parameters:

`timeout` - The period of time in milliseconds to wait for an entry to become available on the queue.

Returns:

the next key

Throws:

ObjectGridException - if an error occurs during processing

AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

WAS XD 6.1

See Also:

QUEUE_TIMEOUT_INFINITE, QUEUE_TIMEOUT_NONE

---

getEntityMetadata

`com.ibm.websphere.projector.md.EntityMetadata` **`getEntityMetadata`**`()`

Retrieve the metadata for the entity associated with this map.

Returns:
the EntityMetadata if an entity is associated with this map or null if there is no entity associated with this map.
Since:
WAS XD 6.1

---

getMapType

`int` **`getMapType`**`()`

Returns the type of the underlying BackingMap.

The return value is equivalent to one of the constants declared on the BackingMap interface, `BackingMap.LOCAL`, `BackingMap.SERVER`, or `BackingMap.CLIENT`.

Returns:
the BackingMap type
Since:
WAS XD 6.1

---

getAgentManager

`com.ibm.websphere.objectgrid.datagrid.AgentManager` **`getAgentManager`**`()`

Returns the Agent Manager that allows DataGrid operations to be performed on the objects within this Map.

This method should only be called on a client ObjectGrid. If called on a non client ObjectGrid an `IllegalStateException` will be thrown

Returns:
AgentManager
Throws:
`IllegalStateException` - if this method is invoked on a non client ObjectGrid
Since:
WAS XD 6.1

---

setLockTimeout

`void` **`setLockTimeout`**`(int seconds)`

Overrides the BackingMap's lock timeout for this ObjectMap.

Establishes the number of seconds that any given fetch (get, getForUpdate, find, findForUpdate) of a cache entry will wait to get a lock. When the lock strategy is `LockStrategy.NONE`, no lock manager is used by this map. In this case, a call to this method does nothing.

Parameters:
`seconds` - is the lock timeout in seconds. The value must be >= -1. A value of -1 is

used to indicate to use the default setting. Use of the constant`USE_DEFAULT` is recommended when "use default" setting is desired. A value of 0 indicates that if a lock cannot be retrieved immediately to time out without waiting for any period of time for the lock to be released and made available.

Throws:

`IllegalArgumentException` - if seconds argument is less than -1 (USE_DEFAULT)

Since:

WAS XD 6.1

See Also:

`USE_DEFAULT`, `BackingMap.setLockTimeout(int)`, `BackingMap.setLockStrategy(LockStrategy)`, `LockStrategy.OPTIMISTIC`, `LockStrategy.PESSIMISTIC`

---

clear

```
void clear()
          throws ObjectGridException
```

Clear all keys from the Map.

This method is an automatic transaction call. The `Session.isTransactionActive()` must answer false prior to invoking this method.

Specification details:

Required permission:	`MapPermission.REMOVE`
Pessimistic read lock acquired:	Map exclusive lock is acquired.
Transaction:	Hybrid. Each shard is cleared in a separate transaction.
Cache tier:	Applies to all cache tiers except the Loader.

Throws:

`ObjectGridException` - if an error occurs during processing
`TransactionAlreadyActiveException` - if a transaction is already started.
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

WAS XD 6.1.0.3

---

lock

```
boolean lock(Object key,
             com.ibm.websphere.objectgrid.LockMode lockMode)
             throws ObjectGridException
```

Obtains a lock for the given key. **Specification details:**

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`, `LockMode.UPGRADABLE` or `LockMode.EXCLUSIVE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

Parameters:

`key` - the key to lock
`lockMode` - the lockMode to obtain for the given key

Returns:

true if the entry exists in the grid or Loader (if one is defined)

Throws:

IllegalArgumentException - if key is `null`
IllegalStateException - if this map is not using LockStrategy.PESSIMISTIC LockStategy
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

---

lockAll

```
List<Boolean> lockAll(List keyList,
                      com.ibm.websphere.objectgrid.LockMode lockMode)
                      throws ObjectGridException
```

Obtains locks for the given keys. **Specification details:**

Required permission:	`MapPermission.READ`
Pessimistic read lock acquired:	`LockMode.SHARED`, `LockMode.UPGRADABLE` or `LockMode.EXCLUSIVE` Note: Transaction isolation is ignored.
Pessimistic locks held:	Yes
Transaction:	Automatic or manual
Cache tier:	Progresses to all tiers until the keys are found and the appropriate locks are acquired.

Parameters:

`keyList` - the keys to lock
`lockMode` - the lockMode to obtain for the given keys

Returns:

a List of `Booleans` indicating whether the entries exists in the grid or Loader (if one is defined)

Throws:

IllegalStateException - if this map is not using LockStrategy.PESSIMISTIC LockStategy
IllegalArgumentException - if keyList is null or contains a `null` element.
ObjectGridException - if an error occurs during processing
AccessControlException - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

---

upsert

```
void upsert(Object key,
            Object value)
            throws ObjectGridException
```

Puts the Object value into the cache at location represented by key.

The value will be pushed down to the BackingMap/Loader at commit time. The semantics of this method are that the Loader will receive a `LogElement.UPSERT` operation and the map will either do an insert or an update to cause the map to contain this updated value. Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

Specification details:

Required permissi on:	`MapPermission.WRITE`
Transact	Automatic or manual

ion:

Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`key` - The entry to insert or update in the map
`value` - The value to insert or update in the map using the key

Throws:

`IllegalArgumentException` - if key is `null`, or if the map does not allow `null` values and value is `null`
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

See Also:

`CopyMode`

---

upsertAll

```
void upsertAll(LinkedHashMap map)
               throws ObjectGridException
```

Puts each of the Object value into the cache at location represented by key contained in the Map. The values will be pushed down to the BackingMap/Loader at commit time. The semantics of this method are that the Loader will receive a `LogElement.UPSERT` operation and the map will either do an insert or an update to cause the map to contain this updated value. Whether or not a copy of the objects is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

Specification details:

Required permission:	`MapPermission.WRITE`
Transaction:	Automatic or manual
Cache tier:	Applied to all tiers during commit. Use `Session.beginNoWriteThrough()` to limit the operation to the Client Cache tier for client maps, or the Server Cache tier for local or shard maps.

Parameters:

`map` - The key/values to be inserted or updated in the map. The type is LinkedHashMap so that the order can be controlled to avoid deadlocks.

Throws:

`IllegalArgumentException` - if map is `null` or contains a `null` key or if `null` values are not allowed and map contains a `null` value.
`ObjectGridException` - if an error occurs during processing
`AccessControlException` - if the Subject or Credential specified on the Session does not have the appropriate permission.

Since:

8.6, XC10 2.5

See Also:

`CopyMode`

---

getKeyOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getKeyOutputFormat`**`()`

Retrieves the data format for all data access APIs that return cache keys for the current session.

Returns:
the data output format or null if the default should be used.

Since:
8.6, XC10 2.5

---

setKeyOutputFormat

`void` **`setKeyOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`
`throws TransactionAlreadyActiveException`

Sets the data format for all data access APIs that return cache keys for the current session.

This method supports map configurations with a `KeyDataSerializer` plug-in defined, or with eXtreme Data Format enabled.

Parameters:
`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default defined on the parent `BackingMap`.

Throws:
`IllegalArgumentException` - thrown when the data format is not valid for the current configuration.
`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

Since:
8.6, XC10 2.5

See Also:
`BackingMap.getKeyOutputFormat()`

---

getValueOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getValueOutputFormat`**`()`

Retrieves the data format for all data access APIs that return cache values for the current session.

Returns:
the data output format or null if the default should be used.

Since:
8.6, XC10 2.5

---

setValueOutputFormat

`void` **`setValueOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`
`throws TransactionAlreadyActiveException`

Sets the data format for all data access APIs that return cache values for the current session.

This method is functionally equivalent to the `setCopyMode(CopyMode, Class)` with the `CopyMode.COPY_TO_BYTES` and `CopyMode.COPY_TO_BYTES_RAW` values when used with a `ValueDataSerializer` or eXtreme Data Format.

This method supports map configurations with a `ValueDataSerializer` plug-in defined, or with eXtreme Data Format enabled.

Parameters:

`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default defined on the parent `BackingMap`.

Throws:

`IllegalArgumentException` - thrown when the data format is not valid for the current configuration.

`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

Since:

8.6, XC10 2.5

---

setPutMode

```
void setPutMode(ObjectMap.PutMode putMode)
                throws TransactionAlreadyActiveException,
                       ObjectGridException
```

Allows the default operation for the `put(Object, Object)` and `putAll(Map)` methods to be changed, allowing the `put` to use the optimized `upsert(Object, Object)` and `upsertAll(LinkedHashMap)` implementations.

The PutMode cannot be changed during a transaction. There must be no active transaction when this method is called.

Parameters:

`putMode` - the mode in which the `put` methods operate.

Throws:

`TransactionAlreadyActiveException` - if a transaction is active and this map has already been used in the transaction.

`ObjectGridException` - if an error occurs during processing

Since:

8.6, XC10 2.5

---

getPutMode

`ObjectMap.PutMode` **`getPutMode`**`()`

Retrieve the current `PutMode` set for this ObjectMap instance.

Returns:

the current `PutMode`.

Since:

8.6, XC10 2.5

---

com.ibm.websphere.objectgrid

Enum ObjectMap.PutMode

java.lang.Object
  java.lang.Enum<ObjectMap.PutMode>
    com.ibm.websphere.objectgrid.ObjectMap.PutMode

All Implemented Interfaces:

Serializable, Comparable<ObjectMap.PutMode>

Enclosing interface:

ObjectMap

```
public static enum ObjectMap.PutMode
extends Enum<ObjectMap.PutMode>
```

Identifies the operation mode of the ObjectMap.put(Object, Object), ObjectMap.putAll(Map) , JavaMap.put(Object, Object) and JavaMap.putAll(Map) methods.

Since:

8.6, XC10 2.5

See Also:

ObjectMap.setPutMode(PutMode)

Enum Constant Summary

INSERTUPDATE **Deprecated.** *Deprecated in 8.6. Use the UPSERT enumeration.*
UPSERT The put API behaves like the upsert method.

Method Summary

static ObjectMap.PutMode	**valueOf**(String name) Returns the enum constant of this type with the specified name.

de	
static ObjectMap.PutMode[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

INSERTUPDATE

```
public static final ObjectMap.PutMode INSERTUPDATE
```

Deprecated. *Deprecated in 8.6. Use the UPSERT enumeration.*
The put API behaves like an insert or update.

UPSERT

```
public static final ObjectMap.PutMode UPSERT
```

The put API behaves like the upsert method.

Method Detail

values

```
public static ObjectMap.PutMode[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (ObjectMap.PutMode c : ObjectMap.PutMode.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

---

valueOf

public static ObjectMap.PutMode **valueOf**(String name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

name - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

IllegalArgumentException - if this enum type has no constant with the specified name

NullPointerException - if the argument is null

---

com.ibm.websphere.objectgrid

Class ObjectGridRuntimeException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        com.ibm.websphere.objectgrid.ObjectGridRuntimeException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

TransactionTimeoutException

```
public class ObjectGridRuntimeException
extends RuntimeException
implements IObjectGridException
```

This exception is the base class for all runtime exceptions thrown by the cache.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridRuntimeException() Constructs a new ObjectGridRuntimeException with `null` as its detail message.
ObjectGridRuntimeException(String message) Constructs a new ObjectGridRuntimeException with the specified detail message.
ObjectGridRuntimeException(String message, Throwable cause) Constructs a new ObjectGridRuntimeException with the specified detail message and cause.
ObjectGridRuntimeException(Throwable cause) Constructs a new ObjectGridRuntimeException with a specified cause.

Method Summary

Throwable	**getCause**() Returns the cause of this ObjectGridRuntimeException or `null` if the cause is nonexistent or unknown.

Throwable	**initCause**(Throwable cause) Initializes the *cause* of this ObjectGridRuntimeException to the specified value.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridRuntimeException

public **ObjectGridRuntimeException**()

Constructs a new ObjectGridRuntimeException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
initCause(Throwable)

ObjectGridRuntimeException

public **ObjectGridRuntimeException**(String message)

Constructs a new ObjectGridRuntimeException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
initCause(Throwable), Throwable.getMessage()

ObjectGridRuntimeException

public **ObjectGridRuntimeException**(Throwable cause)

Constructs a new ObjectGridRuntimeException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ObjectGridRuntimeExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

getCause()

ObjectGridRuntimeException

```
public ObjectGridRuntimeException(String message,
                                  Throwable cause)
```

Constructs a new ObjectGridRuntimeException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridRuntimeException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

getCause(), Throwable.getMessage()

Method Detail

getCause

```
public Throwable getCause()
```

Returns the cause of this ObjectGridRuntimeException or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this ObjectGridRuntimeException to get thrown.)

This implementation returns the cause that was supplied via one of the constructors requiring a `Throwable`, or that was set after creation with the `initCause(Throwable)` method. While it is typically unnecessary to override this method, a subclass can override it to return a cause set by some other means. This is appropriate for a "legacy chained throwable" that predates the addition of chained exceptions to `Throwable`. Note that it is *not* necessary to override any of the `PrintStackTrace` methods, all of which invoke the `getCause` method to determine the cause of an ObjectGridRuntimeException

Specified by:

`getCause` in interface `IObjectGridException`

Overrides:

`getCause` in class `Throwable`

Returns:

the cause of this ObjectGridRuntimeException or `null` if the cause is nonexistent or unknown.

See Also:

ObjectGridRuntimeException(String, Throwable), ObjectGridRuntimeException(Throwable), initCause(Throwable)

---

initCause

```
public Throwable initCause(Throwable cause)
```

Initializes the *cause* of this ObjectGridRuntimeException to the specified value. (The cause is the throwable that caused this ObjectGridRuntimeException to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the ObjectGridRuntimeException. If this

ObjectGridRuntimeException was created with `ObjectGridRuntimeException(Throwable)` or `ObjectGridRuntimeException(String,Throwable)`, this method cannot be called even once.

Specified by:

`initCause` in interface `IObjectGridException`

Overrides:

`initCause` in class `Throwable`

Parameters:

`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)

Returns:

a reference to this `ObjectGridRuntimeException` instance.

Throws:

`IllegalArgumentException` - if `cause` is this ObjectGridRuntimeException. (An ObjectGridRuntimeException cannot be its own cause.)

`IllegalStateException` - if this ObjectGridRuntimeException was created with `ObjectGridRuntimeException(Throwable)` or `ObjectGridRuntimeException(String,Throwable)`, or this method has already been called on this ObjectGridRuntimeException.

See Also:

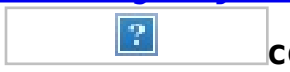
`ObjectGridRuntimeException(String, Throwable)`, `ObjectGridRuntimeException(Throwable)`, `getCause()`

---

com.ibm.websphere.objectgrid

Class ObjectGridManagerFactory

java.lang.Object



com.ibm.websphere.objectgrid.ObjectGridManagerFactory

```
public final class ObjectGridManagerFactory
extends Object
```

This factory class is a high level helper class to get `ObjectGridManager` instances.

Since:

WAS XD 6.0, XC10

Constructor Summary

ObjectGridManagerFactory()

Method Summary

static ObjectGridManager	**getObjectGridManager**() Returns the `ObjectGridManager` singleton.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectGridManagerFactory

public **ObjectGridManagerFactory**()

Method Detail

getObjectGridManager

public static final ObjectGridManager **getObjectGridManager**()

Returns the `ObjectGridManager` singleton.

Returns:
The ObjectGridManager singleton
See Also:
ObjectGridManager

---

com.ibm.websphere.objectgrid

Interface ObjectGridManager

public interface **ObjectGridManager**

ObjectGridManager is responsible for creating or retrieving local ObjectGrid instances and connecting to distributed ObjectGrid servers. Use the `ObjectGridManagerFactory` to retrieve an ObjectGridManager.

Use the `createObjectGrid` methods to create a local, in-memory ObjectGrid instance. The `createObjectGrid` methods give you the choice of caching the created ObjectGrid instance. If you choose to cache the instance, you cannot create an ObjectGrid with the same name unless you remove the previously created ObjectGrid using the `removeObjectGrid(String)` method. A cached `ObjectGrid` instance can later be retrieved using the `getObjectGrid(String)` method.

An example of creating a local in-memory ObjectGrid programmatically:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid grid = ogMgr.createObjectGrid("LocalBookStoreGrid");
grid.defineMap("Orders");
grid.defineMap("Books");
grid.initialize();
...
grid.destroy();
```

An example of creating a local in-memory ObjectGrid using an ObjectGrid descriptor XML file:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
URL objectgridXML  = Thread.currentThread().getContextClassLoader().getResource("configs/objectgr
id.xml");
ObjectGrid grid = ogMgr.createObjectGrid("LocalBookStoreGrid", objectgridXML);
grid.initialize();
...
ogMgr.destroy();
```

Use the `connect` methods to connect to a distributed ObjectGrid. The connect methods return a ClientClusterContext that can then be passed to one of the `getObjectGrid` methods, which will in turn retrieve a client ObjectGrid instance.

An example to connect to a dynamic, distributed ObjectGrid using a catalog server cluster:

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ClientClusterContext ccc = ogMgr.connect("catserver1:2809,catserver2:2809", null, null);
ObjectGrid grid = ogMgr.getObjectGrid(ccc, "BookStoreGrid");
...
ogMgr.disconnect(ccc);
```

An example to connect to an embedded ObjectGrid server (a server running in the current process):

```
ObjectGridManager ogMgr = ObjectGridManagerFactory.getObjectGridManager();
ClientClusterContext ccc = ogMgr.connect((ClientSecurityConfiguration) null, (URL) null);
ObjectGrid grid = ogMgr.getObjectGrid(ccc, "BookStoreGrid");
...
```

```
ogMgr.disconnect(ccc);
```

This interface also allows ObjectGrid trace to be disabled completely for performance improvements especially on a processor with a smaller L2 cache.

Since:
WAS XD 6.0, XC10

Method Summary

ClientClusterContext	**connect**(ClientSecurityConfiguration securityProps, URL overRideObjectGridXml) Use this method to connect to an embedded ObjectGrid server.
ClientClusterContext	**connect**(String catalogServerEndpoints, ClientSecurityConfiguration securityProps, URL overRideObjectGridXml) Connects a client process to a remote ObjectGrid catalog server or catalog server cluster (a dynamic ObjectGrid deployment topology).
ObjectGrid	**createObjectGrid**() Creates a local, in-memory instance of an `ObjectGrid`.
ObjectGr	**createObjectGrid**(String objectGridName) Creates and caches a local, in-memory ObjectGrid instance with the specified name.

Object	**createObjectGrid**(String objectGridName, boolean cacheInstance) Creates a local, in-memory instance of an ObjectGrid with the specified name.
Object	**createObjectGrid**(String objectGridName, URL xmlFile) Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.
Object	**createObjectGrid**(String objectGridName, URL xmlFile, boolean cacheInstance) Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.
Object	**createObjectGrid**(String objectGridName, URL xmlFile, boolean enableXmlValidation, boolean cacheInstance) **Deprecated.** *Deprecated in version 8.6. XML validation is always enabled. Use instead.*
List	**createObjectGrids**(URL xmlFile) Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.
List	**createObjectGrids**(URL xmlFile, boolean cacheInstances) Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.
List	**createObjectGrids**(URL xmlFile, boolean enableXmlValidation, boolean cacheInstances) **Deprecated.** *Deprecated in version 8.6. XML validation is always enabled. Use createObjectGrids(URL, boolean) instead.*
boolean	**disconnect**(ClientClusterContext context) Disconnects a client process from an ObjectGrid server
Ca	

CatalogDomainManager	**getCatalogDomainManager**() Retrieve the CatalogDomainManager for this ObjectGridManager.
ObjectGrid	**getObjectGrid**(ClientClusterContext context, String objectGridName) Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.
ObjectGrid	**getObjectGrid**(ClientClusterContext context, String objectGridName, ObjectGridConfiguration overrideOGConfig) Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.
ObjectGrid	**getObjectGrid**(String objectGridName) Returns a cached local, in-memory `ObjectGrid` instance by name.
List	**getObjectGrids**() Gets a `List` of the local, in-memory `ObjectGrid` instances that have been previously cached.
Map	**getOverrideObjectGridConfigurations**() **Deprecated.** *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*
String	**getTraceSpecification**() Retrieves the trace specification for the current JVM.
void	**putOverrideObjectGridConfigurations**(String clusterName, List objectGridConfigList) **Deprecated.** *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

void	**removeObjectGrid**(String objectGridName) Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances.
void	**removeObjectGrid**(String objectGridName, boolean destroy) Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances and optionally destroys its associated resources.
void	**setOverrideObjectGridConfigurations**(Map overrideMap) **Deprecated.** *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*
void	**setTraceEnabled**(boolean enabledFlag) Enables or disables ObjectGrid trace for the JVM.
void	**setTraceFileName**(String traceFileName) Sets the trace output to go to a file instead of System.out.
void	**setTraceSpecification**(String traceSpec) Set the trace specification for the current JVM.

Method Detail

createObjectGrid

ObjectGrid **createObjectGrid**()
throws ObjectGridException

Creates a local, in-memory instance of an ObjectGrid.

The created ObjectGrid returned by this method is assigned a unique name is **not** cached by the ObjectGridManager. Use the ObjectGrid.setName(String) method to change the ObjectGrid name.

Returns:
an instance of ObjectGrid with a unique name assigned
Throws:
ObjectGridException - if any error occurs during the ObjectGrid creation
See Also:
ObjectGrid, ObjectGrid.setName(String)

---

createObjectGrid

ObjectGrid **createObjectGrid**(String objectGridName,
boolean cacheInstance)
throws ObjectGridException

Creates a local, in-memory instance of an ObjectGrid with the specified name.

The instance of ObjectGrid created can optionally be cached. If an ObjectGrid with the specified name was previously created and cached, an ObjectGridException will be thrown.

Parameters:

`objectGridName` - the name of the ObjectGrid to be created. Must not be `null`.
`cacheInstance` - true, if the ObjectGrid instance should be cached

Returns:

an ObjectGrid instance with the specified name.

Throws:

`IllegalArgumentException` - if objectGridName is `null`
`ObjectGridException` - if an ObjectGrid with this name has already been cached or any error occurs during the ObjectGrid creation.

See Also:

`ObjectGrid`

---

createObjectGrid

```
ObjectGrid createObjectGrid(String objectGridName)
                            throws ObjectGridException
```

Creates and caches a local, in-memory ObjectGrid instance with the specified name.

The ObjectGrid instance created by this method will be cached. Invoking this method is equivalent to invoke `createObjectGrid(String, true)`

Parameters:

`objectGridName` - the Name of the ObjectGrid instance to be created. Must not be `null`.

Returns:

an ObjectGrid instance with the specified name

Throws:

`IllegalArgumentException` - if objectGridName is `null`
`ObjectGridException` - if an ObjectGrid with this name has already been cached, or any error occurs during the ObjectGrid creation

See Also:

`createObjectGrid(String, boolean)`, `ObjectGrid`

---

createObjectGrid

```
@Deprecated
ObjectGrid createObjectGrid(String objectGridName,
                                    URL xmlFile,
                                    boolean enableXmlValidation,
                                    boolean cacheInstance)
                            throws ObjectGridException
```

Deprecated. *Deprecated in version 8.6. XML validation is always enabled. Use `instead.`*

Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.

An ObjectGrid instance is created for the ObjectGrid configuration in the XML file corresponding to the specified ObjectGrid name. If the specified ObjectGrid name cannot be found in the XML file, an exception will be thrown.

This returned ObjectGrid instance optionally can be cached.

If the URL is null, it will be simply ignored. In this case, this method behaves the same as the `createObjectGrid(String, boolean)`.

Parameters:

`objectGridName` - the Name of the ObjectGrid instance to be returned. Must not be `null`.
`xmlFile` - a URL to a well formed xml file based on the ObjectGrid schema.
`enableXmlValidation` - if true the XML is validated
`cacheInstance` - a boolean value indicating whether the returned ObjectGrid instance

defined in the XML will be cached or not. If true, the instance will be cached.

Returns:

an ObjectGrid instance

Throws:

`IllegalArgumentException` - if objectGridName is `null`

`ObjectGridException` - if an ObjectGrid with the same name has been previously cached, an ObjectGrid configuration with the specified name can be found in the xml file, or any other error occurs during ObjectGrid creation.

See Also:

`createObjectGrid(String, boolean)`, `ObjectGrid`

---

createObjectGrid

```
ObjectGrid createObjectGrid(String objectGridName,
                            URL xmlFile,
                            boolean cacheInstance)
                            throws ObjectGridException
```

Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.

An ObjectGrid instance is created for the ObjectGrid configuration in the XML file corresponding to the specified ObjectGrid name. If the specified ObjectGrid name cannot be found in the XML file, an exception will be thrown.

This returned ObjectGrid instance optionally can be cached.

If the URL is null, it will be simply ignored. In this case, this method behaves the same as the `createObjectGrid(String, boolean)`.

Parameters:

`objectGridName` - the Name of the ObjectGrid instance to be returned. Must not be `null`.

`xmlFile` - a URL to a well formed xml file based on the ObjectGrid schema.

`cacheInstance` - a boolean value indicating whether the returned ObjectGrid instance defined in the XML will be cached or not. If true, the instance will be cached.

Returns:

an ObjectGrid instance

Throws:

`IllegalArgumentException` - if objectGridName is `null`

`ObjectGridException` - if an ObjectGrid with the same name has been previously cached, an ObjectGrid configuration with the specified name can be found in the xml file, or any other error occurs during ObjectGrid creation.

Since:

8.6, XC10 2.5

See Also:

`createObjectGrid(String, boolean)`, `ObjectGrid`

---

createObjectGrids

```
List createObjectGrids(URL xmlFile,
                       boolean enableXmlValidation,
                       boolean cacheInstances)
                       throws ObjectGridException
```

Deprecated. *Deprecated in version 8.6. XML validation is always enabled. Use `createObjectGrids(URL, boolean)` instead.*

Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.

The returned ObjecGrid instances can be cached. An ObjectGridException will be thrown when attempting to cache a newly created ObjectGrid that has the same name as an ObjectGrid that has already been cached.

Parameters:

`xmlFile` - the file that defines an ObjectGrid or multiple ObjectGrids

`cacheInstances` - set to true to cache all ObjectGrid instances created based on the file

Returns:

a list of ObjectGrid instances

Throws:

`ObjectGridException` - if attempting to create and cache an ObjectGrid with the same name as an ObjectGrid that has already been cached, or any other error occurs during ObjectGrid creation

See Also:

`ObjectGrid`

---

createObjectGrids

```
List createObjectGrids(URL xmlFile,
                       boolean cacheInstances)
                       throws ObjectGridException
```

Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.

The returned ObjecGrid instances can be cached. An ObjectGridException will be thrown when attempting to cache a newly created ObjectGrid that has the same name as an ObjectGrid that has already been cached.

Parameters:

`xmlFile` - the file that defines an ObjectGrid or multiple ObjectGrids

`cacheInstances` - set to true to cache all ObjectGrid instances created based on the file

Returns:

a list of ObjectGrid instances

Throws:

`ObjectGridException` - if attempting to create and cache an ObjectGrid with the same name as an ObjectGrid that has already been cached, or any other error occurs during ObjectGrid creation

Since:

8.6, XC10 2.5

See Also:

`ObjectGrid`

---

createObjectGrids

```
List createObjectGrids(URL xmlFile)
                       throws ObjectGridException
```

Creates a `List` of a local, in-memory `ObjectGrid` instances based upon the ObjectGrid configurations in the specified ObjectGrid descriptor XML file.

The XML file will be validated against the schema and each ObjectGrid instance that is created will be cached. An ObjectGridException will be thrown when attempting to cache a newly created ObjectGrid that has the same name as an ObjectGrid that has already been cached. Using this method is equivalent to calling the `createObjectGrids(URL, true, true)` method.

Parameters:

`xmlFile` - The XML file to process. ObjectGrid(s) will be created based on the configurations what is in the file.

Returns:

A list of ObjectGrid instances that have been created.

Throws:

`ObjectGridException` - if attempting to create and cache an ObjectGrid with the same name as an ObjectGrid that has already been cached, or any other error occurs during ObjectGrid creation

See Also:

`createObjectGrids(URL, boolean, boolean)`, `ObjectGrid`

---

createObjectGrid

```
ObjectGrid createObjectGrid(String objectGridName,
                            URL xmlFile)
                     throws ObjectGridException
```

Creates a local, in-memory ObjectGrid instance based on the specified ObjectGrid name and the ObjectGrid XML configuration file.

If there is no ObjectGrid with this name defined in the XML file, an ObjectGridException will be thrown. The XML file will be validated against the schema and the ObjectGrid instance created will be cached. Using this method is equivalent to calling the `createObjectGrid(String, URL, true, true)` method.

Parameters:

`objectGridName` - name of the ObjectGrid to create. This ObjectGrid should be defined in the XML file. Must not be `null`.

`xmlFile` - the XML file to process

Returns:

A newly created ObjectGrid

Throws:

`IllegalArgumentException` - if objectGridName is `null`

`ObjectGridException` - if an ObjectGrid with the same name has been previously cached, an ObjectGrid configuration with the specified name can be found in the xml file, or any other error occurs during ObjectGrid creation.

See Also:

`createObjectGrid(String, URL, boolean, boolean)`, `ObjectGrid`

---

removeObjectGrid

```
void removeObjectGrid(String objectGridName)
                      throws ObjectGridException
```

Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances.

Invoking this method is equivalent to calling `removeObjectGrid(String, false)`

Parameters:

`objectGridName` - the name of the ObjectGrid instance to remove from the cache

Throws:

`ObjectGridException` - if an ObjectGrid with the objectGridName was not found in the cache

See Also:

`removeObjectGrid(String, boolean)`

---

removeObjectGrid

```
void removeObjectGrid(String objectGridName,
                      boolean destroy)
                      throws ObjectGridException
```

Removes a local, in-memory ObjectGrid from the cache of ObjectGrid instances and optionally destroys its associated resources.

Parameters:

`objectGridName` - the name of the ObjectGrid instance to remove from the cache
`destroy` - if true, destroy the objectgrid instance and its associated resources

Throws:

`ObjectGridException` - if an ObjectGrid with the objectGridName was not found in the cache

See Also:

`ObjectGrid.destroy()`

---

getObjectGrids

`List` **`getObjectGrids`**`()`

Gets a `List` of the local, in-memory `ObjectGrid` instances that have been previously cached.

This method returns `null` if no ObjectGrid instances have been cached.

Returns:

a list of ObjectGrid instances that have been previously cached or `null` if there are no cached ObjectGrid instances

---

getObjectGrid

`ObjectGrid` **`getObjectGrid`**`(``String` `objectGridName)

Returns a cached local, in-memory `ObjectGrid` instance by name.

This method returns `null` if no `ObjectGrid` with the specified name is currently cached.

Parameters:

`objectGridName` - the cached objectgrid name.

Returns:

a cached ObjectGrid which currently exists.

---

setTraceSpecification

`void` **`setTraceSpecification`**`(``String` `traceSpec)

Set the trace specification for the current JVM.

This operation is a replace operation, not an append operation. The specification should be of the form:

```
 TraceString := <ComponentString>(:<ComponentString>)* ComponentString :=  <ComponentName>=<t
ype>=<state>(,<type>=<state>)*
 ComponentName := a java String state := [enabled|disabled] type := [all|debug|event|entryExi
t]
```

For example, ObjectGrid=all=enabled

Parameters:

`traceSpec` - the new trace specification

---

getTraceSpecification

`String` **`getTraceSpecification`**`()`

Retrieves the trace specification for the current JVM.

Since:
7.1
See Also:
setTraceSpecification(String)

---

setTraceFileName

```
void setTraceFileName(String traceFileName)
```

Sets the trace output to go to a file instead of `System.out`.
The supplied file name can be relative to the working directory or a fully-qualified file name.

Parameters:
`traceFileName` - Name of trace file

---

setTraceEnabled

```
void setTraceEnabled(boolean enabledFlag)
```

Enables or disables ObjectGrid trace for the JVM.

Disabling trace improves the performance when ObjectGrid runs on processors whose L2 caches are not large enough to contain the trace enabled code paths. If this is set to false, ObjectGrid trace is suppressed even if it is enabled using `setTraceSpecification(String)`. By default ObjectGrid trace is enabled.

Parameters:
`enabledFlag` - true to enable trace
Since:
WAS XD 6.0.1
See Also:
setTraceSpecification(String)

---

connect

```
ClientClusterContext connect(ClientSecurityConfiguration securityProps,
                             URL overRideObjectGridXml)
                             throws ConnectException
```

Use this method to connect to an embedded ObjectGrid server. An embedded ObjectGrid server is typically started in an application server process such as IBM WebSphere Application Server. This method allows connecting to the in-memory ObjectGrid server instance without specifying connection information.

This method can be used to connect to both dynamic and static ObjectGrid server deployments.

Parameters:
`securityProps` - client security configuration. The value can be `null` if not running in secure mode.
`overRideObjectGridXml` - This parameter can be `null`. If it is not `null`, the client side configuration of the ObjectGrid plugins are overridden with the ObjectGrid configuration using this `URL`. If this parameter is `null`, client side ObjectGrid plugins can be overridden by providing an overrideMap to `setOverrideObjectGridConfigurations(Map)` or by calling `putOverrideObjectGridConfigurations(String, List)`.

In cases where this parameter is not null, and overriding configuration objects have been provided to the ObjectGridManager by `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)`, the configuration based on the XML file will be used to override ObjectGrid settings. Overriding objects provided to `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)` will be ignored.

Not all plugins can be overridden. For details please see the ObjectGrid documentation.

Returns:

a ClientClusterContext representing the cluster ObjectGrid configuration to which the client is connected.

Throws:

`ConnectException` - if any error occurs connecting to the server

Since:

WAS XD 6.0.1

See Also:

`ClientClusterContext`, `ClientSecurityConfiguration`

---

connect

```
ClientClusterContext connect(String catalogServerEndpoints,
                             ClientSecurityConfiguration securityProps,
                             URL overRideObjectGridXml)
                             throws ConnectException
```

Connects a client process to a remote ObjectGrid catalog server or catalog server cluster (a dynamic ObjectGrid deployment topology). The result ClientClusterContext can then be used to get any ObjectGrid reference managed by that catalog server cluster.

Parameters:

`catalogServerEndpoints` - A concatenated list of host/port pairs belonging to the catalog servers in the form "host:port<,host:port>". This list can be arbitrarily long and is used for bootstrapping only. The first viable address will be used.
`securityProps` - This parameter may be null if the client does not wish to establish a secure connection with the server.
`overRideObjectGridXml` - This parameter can be `null`. If it is not `null`, the client side configuration of the ObjectGrid plugins are overridden with the ObjectGrid configuration using this `URL`. If this parameter is `null`, client side ObjectGrid plugins can be overridden by providing an overrideMap to `setOverrideObjectGridConfigurations(Map)` or by calling `putOverrideObjectGridConfigurations(String, List)`.

In cases where this parameter is not null, and overriding configuration objects have been provided to the ObjectGridManager by `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)`, the configuration based on the XML file will be used to override ObjectGrid settings. Overriding objects provided to `setOverrideObjectGridConfigurations(Map)` or `putOverrideObjectGridConfigurations(String, List)` will be ignored.

Not all plugins can be overridden. For details please see the ObjectGrid documentation.

Returns:

a ClientClusterContext representing the cluster ObjectGrid configuration to which the client is connected.

Throws:

`ConnectException` - If there is a problem connecting to the addresses given.

---

disconnect

```
boolean disconnect(ClientClusterContext context)
```

Disconnects a client process from an ObjectGrid server

Parameters:

`context` - the `ClientClusterContext` object returned from a previous call to one of the `connect` methods The caller must guarantee this parameter is not `null`.

Returns:

`true` if the disconnect was successful, `false` if the supplied context was not connected

Throws:

`IllegalArgumentException` - if the ClientClusterContext is null

Since:

WAS XD 6.0.1

See Also:

`ClientClusterContext`

---

getObjectGrid

```
ObjectGrid getObjectGrid(ClientClusterContext context,
                         String objectGridName)
```

Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.

This method is equivalent to calling `getObjectGrid(context, objectGridName, null)`

Parameters:

`context` - the `ClientClusterContext` object returned from a previous call to one of the `connect` methods The caller must guarantee this parameter is not `null`.

`objectGridName` - the name of the requested client ObjectGrid

Returns:

a client ObjectGrid instance corresponding to a remote ObjectGrid

Throws:

`IllegalArgumentException` - if either provided parameter is null

`ObjectGridRuntimeException` - is the `ObjectGrid` with the specified name is not hosted in any eXtreme Scale servers managed by the catalog server

Since:

WAS XD 6.0.1

See Also:

`ClientClusterContext`, `ObjectGrid`, `getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)`

---

getObjectGrid

```
ObjectGrid getObjectGrid(ClientClusterContext context,
                         String objectGridName,
                         ObjectGridConfiguration overrideOGConfig)
```

Returns a client `ObjectGrid` instance corresponding to an `ObjectGrid` in an ObjectGrid cluster.

This method replaces the get/set/putOverrideObjectGridConfigurations methods. Those methods had thread safety issues. In addition they were global in nature so we end up having configuration override happen for all client connections unless it was managed correctly. If ClientClusterContext was used previously to get an ObjectGrid for the given name, the same ObjectGrid instance is returned even if the `overrideOGConfig` parameter is different.

Parameters:

`context` - the `ClientClusterContext` object returned from a previous call to one of the `connect` methods The caller must guarantee this parameter is not `null`.
`objectGridName` - the name of the requested client ObjectGrid
`overrideOGConfig` - This parameter can be `null`. If it is not `null`, the client side configuration of the ObjectGrid plugins are overridden with the `ObjectGridConfiguration` provided. The provided override configuration takes precedence over any other provided override configuration for the requested ObjectGrid name provided by the `connect`, `putOverrideObjectGridConfigurations`, and `putOverrideObjectGridConfigurations` methods.

Not all plugins can be overridden. For details please see the ObjectGrid documentation.

Returns:

a client ObjectGrid instance corresponding to a remote ObjectGrid

Since:

7.1.1

See Also:

`ObjectGridConfiguration`, `ObjectGridConfigFactory`

---

setOverrideObjectGridConfigurations

`void` **`setOverrideObjectGridConfigurations`**`(Map overrideMap)`

Deprecated. *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

Override ObjectGrid settings on client side ObjectGrids by passing in a Map where each key corresponds to a cluster name or domain name and each value is a `List` of `ObjectGridConfiguration` objects to be overridden.

Client side configuration of ObjectGrid and BackingMap plugins are overridden using the ObjectGridConfiguration values provided in the List. To override a Plugin, each `ObjectGridConfiguration` object must have a name that matches the name of the ObjectGrid to be overridden. `BackingMapConfiguration` objects must have the same name as a BackingMap and be associated with the proper`ObjectGridConfiguration`.

Not all plugins can be overridden. ObjectGrid plugins that can be overridden on the client side are TransactionCallback and ObjectGridEventListener. BackingMap plugins that can be overridden on the client side are Evictor and MapEventListener. Settings for the builtin Evictor can also be altered on the BackingMap. These settings include numberOfBuckets, timeToLive, and ttlEvictorType.

Parameters:

`overrideMap` - a Map that will be used to override ObjectGrid settings on the client side. To override client side settings, each key of the Map must be a String with a value that corresponds to a cluster name or domain name. Each value of the overrideMap must be a `java.util.List`. The List elements must be `ObjectGridConfiguration` objects. Each call to a connect method with a clusterName that matches a key in the overrideMap will result in the client side settings being overridden using the List of ObjectGridConfiguration objects provided in the key's corresponding value. Pass in `null` to clear an overrideMap that was previously set and thereby remove any overriding settings from future connect calls.

Throws:

`IllegalArgumentException` - if any keys or values are null or if keys or values are of the wrong type

Since:

WAS XD 6.0.1.2

See Also:

`connect(String, ClientSecurityConfiguration, URL)`, `connect(ClientSecurityConfiguration,`

URL), ObjectGridConfiguration, BackingMapConfiguration, getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)

---

putOverrideObjectGridConfigurations

```
void putOverrideObjectGridConfigurations(String clusterName,
                                         List objectGridConfigList)
```

Deprecated. *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

Put an entry into the Map that is used to override client side ObjectGrid and BackingMap plugins.

Parameters:

`clusterName` - to be used as a key in the Map used to override client side ObjectGrid plugins. If a connect method is called with a matching clusterName, the client side ObjectGrid and BackingMap plugins can be overridden using the objectGridConfigList. In the dynamic environment, use the domain name to override ObjectGrid settings.

`objectGridConfigList` - a List of `ObjectGridConfiguration` objects that will be used to override client side ObjectGrid settings if a connect method is called with a cluster name that matches the clusterName on this method

Throws:

`IllegalArgumentException` - if the clusterName or objectGridConfigList is null

See Also:

getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)

---

getOverrideObjectGridConfigurations

```
Map getOverrideObjectGridConfigurations()
```

Deprecated. *This method is replaced in 7.1.1 with the getObjectGrid method that takes a ObjectGridConfigurations.*

Get the Map that is used to override client side ObjectGrid and BackingMap plugins.

Returns:

the Map that was set by the call to `setOverrideObjectGridConfigurations`. The Map may also have entries that were put there using the `putOverrideObjectGridConfigurations` method.

See Also:

getObjectGrid(ClientClusterContext, String, ObjectGridConfiguration)

---

getCatalogDomainManager

```
CatalogDomainManager getCatalogDomainManager()
```

Retrieve the CatalogDomainManager for this ObjectGridManager.

Returns:

the CatalogDomainManager, if available. Returns null if a CatalogDomainManager is not supported in the current runtime environment.

Since:

8.5

---

com.ibm.websphere.objectgrid

Class ObjectGridException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

CacheEntryException, ConnectException, ContinuousQueryException, DuplicateKeyException, KeyNotFoundException, LoaderException, LockException, NoActiveTransactionException, ObjectGridConfigurationException, ObjectGridSecurityException, ReadOnlyException, TransactionCallbackException, TransactionException, UndefinedMapException

---

```
public class ObjectGridException
extends Exception
implements IObjectGridException
```

Base exception class for all checked exceptions thrown by the ObjectGrid product.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

ObjectGridException() Constructs a new ObjectGridException with `null` as its detail message.
ObjectGridException(String message) Constructs a new ObjectGridException with the specified detail message.
ObjectGridException(String message, Throwable cause) Constructs a new ObjectGridException with the specified detail message and cause.
ObjectGridException(Throwable cause) Constructs a new ObjectGridException with a specified cause.

Method Summary

Throwable	**getCause**() Returns the cause of this ObjectGridException or `null` if the cause is nonexistent or unknown.

le	
Throwable	**initCause**(Throwable cause) Initializes the *cause* of this ObjectGridException to the specified value.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectGridException

public **ObjectGridException**()

Constructs a new ObjectGridException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
initCause(Throwable)

ObjectGridException

public **ObjectGridException**(String message)

Constructs a new ObjectGridException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
initCause(Throwable), Throwable.getMessage()

ObjectGridException

public **ObjectGridException**(Throwable cause)

Constructs a new ObjectGridException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ObjectGridExceptions that are little more than wrappers for other throwables.

Parameters:
`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:
getCause()

ObjectGridException

```
public ObjectGridException(String message,
                           Throwable cause)
```

Constructs a new ObjectGridException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ObjectGridException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:
getCause(), Throwable.getMessage()

Method Detail

getCause

```
public Throwable getCause()
```

Returns the cause of this ObjectGridException or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this ObjectGridException to get thrown.)

This implementation returns the cause that was supplied via one of the constructors requiring a `Throwable`, or that was set after creation with the `initCause(Throwable)` method. While it is typically unnecessary to override this method, a subclass can override it to return a cause set by some other means. This is appropriate for a "legacy chained throwable" that predates the addition of chained exceptions to `Throwable`. Note that it is *not* necessary to override any of the `PrintStackTrace` methods, all of which invoke the `getCause` method to determine the cause of an ObjectGridException

Specified by:
`getCause` in interface `IObjectGridException`
Overrides:
`getCause` in class `Throwable`

Returns:
the cause of this ObjectGridException or `null` if the cause is nonexistent or unknown.
See Also:
ObjectGridException(String, Throwable), ObjectGridException(Throwable), initCause(Throwable)

---

initCause

```
public Throwable initCause(Throwable cause)
```

Initializes the *cause* of this ObjectGridException to the specified value. (The cause is the throwable that caused this ObjectGridException to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the ObjectGridException. If this

ObjectGridException was created with `ObjectGridException(Throwable)` or `ObjectGridException(String,Throwable)`, this method cannot be called even once.

Specified by:

`initCause` in interface `IObjectGridException`

Overrides:

`initCause` in class `Throwable`

Parameters:

`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)

Returns:

a reference to this `ObjectGridException` instance.

Throws:

`IllegalArgumentException` - if `cause` is this ObjectGridException. (An ObjectGridException cannot be its own cause.)

`IllegalStateException` - if this ObjectGridException was created with `ObjectGridException(Throwable)` or `ObjectGridException(String,Throwable)`, or this method has already been called on this ObjectGridException.

See Also:

`ObjectGridException(String, Throwable)`, `ObjectGridException(Throwable)`, `getCause()`

---

com.ibm.websphere.objectgrid

Interface ObjectGrid

public interface **ObjectGrid**

This object is used for creating sessions to the ObjectGrid. It is the central core of the ObjectGrid framework. Besides creating `Sessions`, it is also responsible for defining `BackingMaps`, setting a `TransactionCallback`, adding event listeners, and managing the security settings.

Since:

WAS XD 6.0, XC10

Field Summary

static int	**CLIENT** Indicates the ObjectGrid is a client-side distributed ObjectGrid.
static int	**DEFAULT_TX_TIMEOUT_VALUE** The default transaction time out value of 10 minutes if no transaction time out value is set.
static int	**LOCAL** Indicates the ObjectGrid is a local ObjectGrid.
static int	**SERVER** Indicates the ObjectGrid is a server-side distributed ObjectGrid.

Method Summary

void	**addEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener) Adds an EventListener.
void	**addEventListener**(ObjectGridEventListener listener) **Deprecated.** *This method is deprecated in version 7.1.1, use the addEventListener(EventListener) method.*
BackingMap	**createMap**(String name) Creates a BackingMap, but does not associate it with this ObjectGrid.
BackingMap	**defineMap**(String name) Defines a BackingMap that will be used by the application.
void	**destroy**() Destroys this instance.
com.ibm.websphere.objectgrid.revision	**getCollisionArbiter**() Retrieves the CollisionArbiter that this grid is using to resolve revision collisions.

CollisionArbiter	
List	**getEventListeners**() Returns the current list of EventListeners.
List	**getListOfMapNames**() Gets the list of map names currently defined for this ObjectGrid instance.
BackingMap	**getMap**(String name) Returns a BackingMap previously configured by calling the defineMap(String) or setMaps(List) method.
String	**getName**() Gets the name of this ObjectGrid.
int	**getObjectGridType**() Returns the type of ObjectGrid.
Session	**getSession**() Gets a Session object that can be used by a single thread at a time.
Session	**getSession**(CredentialGenerator credGen) Get a session using a CredentialGenerator.
Session	**getSession**(Subject subject) Allows the use of a specific Subject rather than use the SubjectSource configured on the ObjectGrid to get a Session.

com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener.State	**getState**() Retrieve the current life cycle state of this ObjectGrid.
Trans	

ActionCallback	**getTransactionCallback**() Retrieves the TransactionCallback object.
int	**getTxIsolation**() Retrieves the default transaction isolation level.
int	**getTxTimeout**() Gets transaction timeout setting for this ObjectGrid instance.
void	**initialize**() Begins the bootstrapping of the ObjectGrid and Session instances.
boolean	**isSecurityEnabled**() Checks whether security is enabled on this ObjectGrid or not.
void	**registerEntities**(Class[] entities) Register one or more entities based on the class metadata.
void	**registerEntities**(URL entityXML) Registers one ore more entities from an entity XML file.
void	**removeEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener) Removes an EventListener.
void	**removeEventListener**(ObjectGridEventListener listener) **Deprecated.** *This method is deprecated in version 7.1.1, use the removeEventListener(EventListener) method.*
int	**reserveSlot**(String containerName) Allows plugins on this ObjectGrid to reserve slots for use to store context data.
void	**setAccessByCreatorOnlyMode**(int accessByCreatorOnlyMode) Set the "access by creator only" mode.
void	**setAuthorizationMechanism**(int authMechanism) Sets the authorization mechanism.
v	**setCollisionArbiter**(com.ibm.websphere.objectgrid.revision.CollisionArbiter arbiter)

void	Sets the CollisionArbiter that is responsible for arbitration of revision conflicts.
void	**setEventListeners**(List listeners) **Deprecated.** *This method is deprecated in version 7.1.1. Use the addEventListener(EventListener) or removeEventListener(EventListener) methods. Plug-ins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*
void	**setMaps**(List mapList) Clears any BackingMaps that have been previously defined on this ObjectGrid and replaces them with the List of BackingMaps provided.
void	**setName**(String gridName) Sets the name of this ObjectGrid.
void	**setObjectGridAuthorization**(com.ibm.websphere.objectgrid.security.plugins.ObjectGridAuthorization ogAuthorization) Sets the ObjectGridAuthorization for this ObjectGrid instance.
void	**setPermissionCheckPeriod**(int period) Sets the permission check period.
void	**setQueryConfig**(QueryConfig queryConfig) Set the QueryConfig object for this ObjectGrid.
void	**setSecurityEnabled**() Enables the ObjectGrid security.
void	**setSubjectSource**(com.ibm.websphere.objectgrid.security.plugins.SubjectSource source) Sets the SubjectSource plugin.
void	**setSubjectValidation**(com.ibm.websphere.objectgrid.security.plugins.SubjectValidation subjectValidation) Sets the SubjectValidation for this ObjectGrid instance.
void	**setTransactionCallback**(TransactionCallback callback) Sets the TransactionCallback object.
void	**setTxIsolation**(int level) Sets the default transaction isolation level for all sessions created by the ObjectGrid.
void	**setTxTimeout**(int timeout) Sets the transaction timeout value to a specified number of seconds.

Field Detail

DEFAULT_TX_TIMEOUT_VALUE

```
static final int DEFAULT_TX_TIMEOUT_VALUE
```

The default transaction time out value of 10 minutes if no transaction time out value is set.

Since:
WXS 7.1.0.0 FIX1

See Also:
Constant Field Values

LOCAL

```
static final int LOCAL
```

Indicates the ObjectGrid is a local ObjectGrid.

See Also:
Constant Field Values

SERVER

```
static final int SERVER
```

Indicates the ObjectGrid is a server-side distributed ObjectGrid.

See Also:
Constant Field Values

CLIENT

```
static final int CLIENT
```

Indicates the ObjectGrid is a client-side distributed ObjectGrid.

See Also:
Constant Field Values

Method Detail

getSession

```
Session getSession()
                  throws ObjectGridException,
                         TransactionCallbackException
```

Gets a Session object that can be used by a single thread at a time.

It is not allowed to share this `Session` object between threads without placing a critical section around it. While the core framework allows the object to move between threads, the `TransactionCallback` and `Loader` may prevent this usage, especially in J2EE environments.

When the ObjectGrid is a local ObjectGrid, and its security is enabled, this method will use the `SubjectSource` to get a `Subject` object and then associate the `Subject` object with this session .

When the ObjectGrid is a distributed ObjectGrid (client server mode), and its security is

enabled, this method will utilize the client server security infrastructure to get a secure session.

If the `initialize()` method has not been invoked prior to the first `getSession` invocation, an implicit initialization will occur. This ensures that all of the configuration is complete before any runtime usage is required.

Returns:
An instance of `Session`
Throws:
`ObjectGridException` - if an error occurs during processing
`TransactionCallbackException` - if the `TransactionCallback` throws an exception
`IllegalStateException` - if this method is called after the `destroy()` method is called.
See Also:
`destroy()`, `initialize()`, `Session`, `SubjectSource`

---

getSession

```
Session getSession(Subject subject)
                   throws ObjectGridException,
                          TransactionCallbackException,
                          com.ibm.websphere.objectgrid.security.plugins.InvalidSubjectException
```

Allows the use of a specific `Subject` rather than use the `SubjectSource` configured on the ObjectGrid to get a Session.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided `Subject` object will not be used.

If the `initialize()` method has not been invoked prior to the first `getSession` invocation, an implicit initialization will occur. This ensures that all of the configuration is complete before any runtime usage is required.

Parameters:
`subject` - `Subject` to associate with the returned Session
Returns:
An instance of `Session`
Throws:
`ObjectGridException` - if an error occurs during processing
`TransactionCallbackException` - if the `TransactionCallback` throws an exception
`com.ibm.websphere.objectgrid.security.plugins.InvalidSubjectException` - the subject passed in is invalid based on the `SubjectValidation` mechanism.
`IllegalStateException` - if this method is called after the `destroy()` method is called.
See Also:
`destroy()`, `initialize()`, `Session`, `SubjectValidation`

---

setTransactionCallback

```
void setTransactionCallback(TransactionCallback callback)
```

Sets the TransactionCallback object.

A single cache is a single domain. All `Loader`s defined for `BackingMap`s in an ObjectGrid will normally cooperate, thus a corresponding `TransactionCallback` object needs to be set on the ObjectGrid.

A TransactionCallback that implements the `ObjectGridLifecycleListener` interface is automatically added as if the `addEventListener(EventListener)` method was called. Any previous callback which implements `ObjectGridLifecycleListener` interface is removed as if the `removeEventListener(EventListener)` method was called.

A TransactionCallback may implement the `ObjectGridPlugin` interface in order to receive enhanced ObjectGrid plug-in lifecycle method calls. The plug-in is also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:

`callback` - An instance of a TransactionCallback

Throws:

`IllegalArgumentException` - if callback is `null`

`IllegalStateException` - if this method is called after the `initialize()` method is called.

See Also:

`initialize()`, `TransactionCallback`

---

getTransactionCallback

`TransactionCallback` **`getTransactionCallback`**`()`

Retrieves the TransactionCallback object.

The TransactionCallback can be used in conjunction with the `TxID` to house transaction-specific context data, such as the connection to the database.

Returns:

the argument that was passed to the `setTransactionCallback(TransactionCallback)` method of this interface or a default `TransactionCallback` object if setTransactionCallback was not previously called for this ObjectGrid.

See Also:

`setTransactionCallback(TransactionCallback)`, `TransactionCallback`

---

setCollisionArbiter

`void` **`setCollisionArbiter`**`(com.ibm.websphere.objectgrid.revision.CollisionArbiter arbiter)`

Sets the CollisionArbiter that is responsible for arbitration of revision conflicts.

A CollistionArbiter that implements the `ObjectGridLifecycleListener` interface is automatically added as if the `addEventListener(EventListener)` method was called. Any previous arbiter which implements `ObjectGridLifecycleListener` interface is removed as if the `removeEventListener(EventListener)` method was called.

A CollisionArbiter may implement the `ObjectGridPlugin` interface in order to receive enhanced ObjectGrid plug-in lifecycle method calls. The plug-in is also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:

`arbiter` - The arbitration logic that will be used to resolve collisions.

Since:

7.1

---

getCollisionArbiter

`com.ibm.websphere.objectgrid.revision.CollisionArbiter` **`getCollisionArbiter`**`()`

Retrieves the CollisionArbiter that this grid is using to resolve revision collisions.

Returns:

The arbitration logic that is responsible for resolving revision collisions.

Since:

---

defineMap

[BackingMap](#) **defineMap**([String](#) name)

Defines a BackingMap that will be used by the application.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `initialize()` method. Also, keep in mind that the `getSession` methods implicitly call the `initialize()` method if it has yet to be called by the application.

Parameters:
`name` - the name of the map being defined.
Returns:
a BackingMap reference
Throws:
[IllegalArgumentException](#) - if name is `null`
[IllegalStateException](#) - if this method is called after the `initialize()` method is called.
See Also:
[initialize()](#), [BackingMap](#)

---

createMap

[BackingMap](#) **createMap**([String](#) name)

Creates a BackingMap, but does not associate it with this ObjectGrid.

This method is to be used in tandem with the `setMaps(List)` method, which will associate BackingMaps with this ObjectGrid. These methods are for use when configuring an ObjectGrid with the Spring Framework.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
name - the name of the map being defined.
Returns:
a BackingMap reference
Throws:
[IllegalArgumentException](#) - if name is null
[IllegalStateException](#) - if this method is called after the initialize() method is called.
Since:
WAS XD 6.0.1
See Also:
[initialize()](#), [setMaps(List)](#)

---

setMaps

void **setMaps**([List](#) mapList)

Clears any BackingMaps that have been previously defined on this ObjectGrid and replaces them with the List of BackingMaps provided.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
mapList - a list of BackingMaps to set on this ObjectGrid.
Throws:

IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.0.1

See Also:

createMap(String), initialize()

---

getListOfMapNames

List **getListOfMapNames**()

Gets the list of map names currently defined for this ObjectGrid instance.

Note, once the initialize() method is called, the List returned will not change. However, it could change if called prior to initialization. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Returns:

a List of String objects, one String per map that was previously configured by the defineMap(String) or setMaps(List) method. An empty List is returned if no maps are currently defined.

See Also:

defineMap(String), initialize(), setMaps(List)

---

getMap

BackingMap **getMap**(String name)

Returns a BackingMap previously configured by calling the defineMap(String) or setMaps(List) method.

Parameters:

name - the same name that was used as an argument to the defineMap(String) or createMap(String) method. A null reference is returned if a map is not associated with this ObjectGrid for the specified map name.

Returns:

the BackingMap instance

See Also:

createMap(String), defineMap(String), setMaps(List)

---

initialize

void **initialize**()
throws ObjectGridException

Begins the bootstrapping of the ObjectGrid and Session instances.

After this method has been invoked, the configuration of the ObjectGrid is considered complete and is ready for runtime usage. Any additional configuration method invocations, such as defineMap(String), will result in an exception. This method is considered optional since the first call to one of the getSession methods will perform an implicit initialization.

Throws:

ObjectGridException - if an error occurs during processing

---

addEventListener

void **addEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener)

Adds an EventListener.

Significant events will be communicated to interested listeners through the ObjectGridEventListener and ObjectGridLifecycleListener callback interface. Multiple event listeners are allowed to be registered, with no implied ordering of event notifications.

Note, this method is allowed to be invoked before and after the initialize() method.

Object grid plug-ins (TransactionCallback, CollisionArbiter) that implement the ObjectGridLifecycleListener are automatically added as lifecycle listeners when added to the ObjectGrid.

Parameters:
listener - An instance of ObjectGridEventListener or ObjectGridLifecycleListener
Throws:
IllegalArgumentException - if listener is null or not an instance of ObjectGridEventListener, ObjectGridLifecycleListener.
IllegalStateException - if this method is called during initialization by one of the configured plugins and the ObjectGrid runtime is not in a usable state to initialize the ObjectGridEventListener.
See Also:
ObjectGridEventListener, ObjectGridLifecycleListener, EventListener

---

addEventListener

void **addEventListener**(ObjectGridEventListener listener)

Deprecated. *This method is deprecated in version 7.1.1, use the addEventListener(EventListener) method.*

Provided for compatibility with old releases, use the addEventListener(EventListener) method.

Parameters:
listener -

---

removeEventListener

void **removeEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener listener)

Removes an EventListener.

This method removes an ObjectGridEventListener or ObjectGridLifecycleListener that was previously added to this object using the addEventListener(ObjectGridEventListener) or setEventListeners(List) method. If the desired ObjectGridEventListener is not found, no error will be returned.

Note, this method is allowed to be invoked before and after the initialize() method. Object grid plug-ins (TransactionCallback, CollisionArbiter) that implement the ObjectGridLifecycleListener are automatically removed as lifecycle listeners when removed from the ObjectGrid.

Parameters:
listener - An instance of ObjectGridEventListener or ObjectGridLifecycleListener
Throws:
IllegalArgumentException - if listener is null or not an instance of ObjectGridEventListener, ObjectGridLifecycleListener
See Also:
addEventListener(EventListener), ObjectGridEventListener, EventListener

---

removeEventListener

void **removeEventListener**(ObjectGridEventListener listener)

Deprecated. *This method is deprecated in version 7.1.1, use the removeEventListener(EventListener) method.*

Provided for compatibility with old releases, use the removeEventListener(EventListener) method.

Parameters:
listener -

---

setEventListeners

@Deprecated
void **setEventListeners**(List listeners)

Deprecated. *This method is deprecated in version 7.1.1. Use the addEventListener(EventListener) or removeEventListener(EventListener) methods. Plug-ins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*

This overwrites the current list of EventListeners and replaces it with the supplied List of EventListeners

Note, this method is allowed to be invoked before and after the initialize() method.

Parameters:
listeners - List of ObjectGridEventListeners and ObjectGridLifecycleListener instances
Throws:
ClassCastException - if one of the elements in the provided list is not an instance of ObjectGridEventListener
IllegalArgumentException - if listeners is null, contains a null reference, or contains an instance of a type other than ObjectGridEventListener and ObjectGridLifecycleListener
IllegalStateException - if this method is called during initialization by one of the configured plugins and the ObjectGrid runtime is not in a usable state to initialize the ObjectGridEventListener objects.
See Also:
EventListener, addEventListener(EventListener), removeEventListener(EventListener)

---

getEventListeners

List **getEventListeners**()

Returns the current list of EventListeners.

Returns:
The current list of EventListeners.
See Also:
addEventListener(EventListener), EventListener, ObjectGridEventListener, ObjectGridLifecycleListener

---

getName

String **getName**()

Gets the name of this ObjectGrid.

This method is useful for authorization as all Maps are prefixed with the ObjectGrid name.

Returns:
The name of the ObjectGrid.
See Also:
setName(String)

---

setName

void **setName**(String gridName)

Sets the name of this ObjectGrid. Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
gridName - The ObjectGrid name to use.
Throws:
IllegalArgumentException - if gridName is null
IllegalStateException - if this method is called after the initialize() method is called.

---

reserveSlot

int **reserveSlot**(String containerName)

Allows plugins on this ObjectGrid to reserve slots for use to store context data.

Currently the TxID object is the only object that uses slots for storing context data. TxID slots are used for storing transactional context data.

Once a slot is reserved, the slot assignment is permanent and cannot be given back. Note, this method is allowed to be invoked before and after the initialize() method.

Parameters:
containerName - The name of the Object with the slots. Currently TxID.SLOT_NAME is the only supported value for this argument.
Returns:
The slot index to use.
Throws:
IllegalArgumentException - if containerName is not TxID.SLOT_NAME.
See Also:
TxID.SLOT_NAME, TxID.getSlot(int), TxID.putSlot(int, Object)

---

setSubjectValidation

void **setSubjectValidation**(com.ibm.websphere.objectgrid.security.plugins.SubjectValidation subjectValidation)

Sets the SubjectValidation for this ObjectGrid instance.

Passing null to this method removes a previously set SubjectValidation object from an earlier invocation of this method and indicates that this ObjectGrid is not associated with a SubjectValidation object.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided SubjectValidation object will not be used.

A SubjectValidation plugin can be used to validate the Subject object passed in is a valid Subject. Please refer to SubjectValidation for more details.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
subjectValidation - the SubjectValidation plugin
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
getSession(Subject), initialize(), SubjectValidation

---

setAuthorizationMechanism

void **setAuthorizationMechanism**(int authMechanism)

Sets the authorization mechanism.

If this method is not invoked, the default authorization mechanism is SecurityConstants.AUTHORIZATION_MECHANISM_JAAS.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provide authorization mechanism will not be used.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
authMechanism - the authorization mechanism, must be one of the final static variable on the SecurityConstants class.
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize(), SecurityConstants.AUTHORIZATION_MECHANISM_CUSTOM, SecurityConstants.AUTHORIZATION_MECHANISM_JAAS

setSecurityEnabled

void **setSecurityEnabled**()

Enables the ObjectGrid security.

Security on the ObjectGrid level refers to ObjectGrid authorizations.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize()

---

isSecurityEnabled

boolean **isSecurityEnabled**()

Checks whether security is enabled on this ObjectGrid or not.

Security on the ObjectGrid level refers to ObjectGrid authorizations. Security is disabled by default.

Returns:
true if security is enabled on this ObjectGrid; false otherwise.
See Also:
setSecurityEnabled()

---

setPermissionCheckPeriod

void **setPermissionCheckPeriod**(int period)

Sets the permission check period.

This method takes a single parameter indicating how often the customer wants to check the permission used to allow a client access. If the parameter is 0 then every single authorized operation call will ask the authorization mechanism, either JAAS authorization or custom authorization to check if the current Subject has permission. This approach may be prohibitively expensive from a performance point of view depending on the authorization implementation, but if it is required then you can do it. Alternatively, if the parameter is > 0 then it indicates the number of seconds to cache a set of permissions before returning to the authorization mechanism to refresh them. This mechanism provides much better performance, but you run the risk that if the back-end permissions are changed during this time, the ObjectGrid will possibly allow or prevent access even though the back-end security provider has been modified.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
period - the permission check period in seconds.
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize()

---

setSubjectSource

void **setSubjectSource**(com.ibm.websphere.objectgrid.security.plugins.SubjectSource source)

Sets the SubjectSource plugin.

Passing null to this method removes a previously set SubjectSource object from an earlier invocation of this method and indicates that this ObjectGrid is not associated with a SubjectSource object.

A SubjectSource plugin can be used to get a Subject object from the environment to represent the ObjectGrid client.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided SubjectSource object will not be used.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
source - the SubjectSource plugin
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
See Also:
initialize(), SubjectSource

---

setTxTimeout

void **setTxTimeout**(int timeout)

Sets the transaction timeout value to a specified number of seconds.

Any transaction that is started by use of a Session returned by one of the getSession methods on this interface must complete within the number of seconds specified by the transaction timeout parameter of this method. The timeout value is the maximum number of seconds the transaction is allowed to execute. If a transaction executes longer than this amount, a TransactionTimeoutException is thrown and the transaction is rolled back even if commit is requested.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

The transaction timeout is used by any transaction started by a Session that is returned by the getSession methods of this interface. Since this method must be called prior to getSession method to avoid IllegalStateException, this method only affects transactions that are started after this method is called. If this method is never called, the transaction is allowed unlimited amount of time to complete.

Parameters:
timeout - is the transaction timeout value in seconds. Use a value of 0 to indicate a transaction is allowed unlimited amount of time so that no TransactionTimeoutException ever occurs.
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
Since:
WAS XD 6.0.1
See Also:
initialize(), Session.TRANSACTION_NO_TIMEOUT, Session.setTransactionTimeout(int), TransactionTimeoutException

---

getTxTimeout

int **getTxTimeout**()

Gets transaction timeout setting for this ObjectGrid instance.

Returns:
timeout value that was passed to the setTxTimeout(int) method or 0 if setTxTimeout was never called.
Since:
WAS XD 6.0.1
See Also:

setTxTimeout(int)

---

setTxIsolation

void **setTxIsolation**(int level)

Sets the default transaction isolation level for all sessions created by the ObjectGrid. The constants defined in the Session interface are the possible transaction isolation levels. The default is Session.TRANSACTION_REPEATABLE_READ.

Parameters:
level - one of the following Session constants: Session.TRANSACTION_READ_UNCOMMITTED, Session.TRANSACTION_READ_COMMITTED or Session.TRANSACTION_REPEATABLE_READ
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
IllegalArgumentException - if this method is includes and invalid transaction isolation level.
Since:
7.1.1

---

getTxIsolation

int **getTxIsolation**()

Retrieves the default transaction isolation level.

Returns:
the current transaction isolation level.
Since:
7.1.1
See Also:
setTxIsolation(int)

---

destroy

void **destroy**()

Destroys this instance.

This method should be invoked when the ObjectGrid is no longer being used. When this method is called, the ObjectGrid can free up any resources it is using. No new Sessions can be created or used after the destroy() has been invoked. Any in-flight Sessions will be allowed to continue, if the resources are still available to complete processing.

---

getSession

Session **getSession**(CredentialGenerator credGen)
throws ObjectGridException,
TransactionCallbackException

Get a session using a CredentialGenerator.

This method can only be called by the ObjectGrid client in an ObjectGrid client server environment. If ObjectGrid is used in a local model, that is, within the same JVM with no client or server existing, getSession(Subject) or the SubjectSource plugin should be used to secure the ObjectGrid.

If the initialize() method has not been invoked prior to the first getSession invocation, an implicit initialization will occur. This ensures that all of the configuration is complete before any runtime usage is required.

Parameters:
credGen - A CredentialGenerator for generating a credential for the session returned.
Returns:
An instance of Session
Throws:
ObjectGridException - if an error occurs during processing

TransactionCallbackException - if the TransactionCallback throws an exception
IllegalStateException - if this method is called after the destroy() method is called.

Since:

WAS XD 6.0.1

See Also:

destroy(), initialize(), CredentialGenerator, Session

---

setQueryConfig

void **setQueryConfig**(QueryConfig queryConfig)

Set the QueryConfig object for this ObjectGrid. A QueryConfig object provides query configurations for executing object queries over the maps in this ObjectGrid.

Parameters:

queryConfig - The QueryConfig to associate with this ObjectGrid instance.

Throws:

IllegalArgumentException - if queryConfig is null.
IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.1

See Also:

QueryConfig

---

registerEntities

void **registerEntities**(URL entityXML)

Registers one ore more entities from an entity XML file.

Entity registration is required prior to ObjectGrid initialization to bind an Entity with a BackingMap and any defined indices.

This method may be called multiple times.

Parameters:

entityXML - the URL of the entity XML that defines the entities.

Throws:

IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.1

---

registerEntities

void **registerEntities**(Class[] entities)

Register one or more entities based on the class metadata.

Entity registration is required prior to ObjectGrid initialization to bind an Entity with a BackingMap and any defined indices.

This method may be called multiple times.

Parameters:

entities - one or more annotated entity classes to register as entities.

Throws:

IllegalStateException - if this method is called after the initialize() method is called.

Since:

WAS XD 6.1

---

getObjectGridType

int **getObjectGridType**()

Returns the type of ObjectGrid.

The return value is equivalent to one of the constants declared on this interface, LOCAL, SERVER, or CLIENT.

Returns:
the ObjectGrid type
Since:
WAS XD 6.1

---

setObjectGridAuthorization

void **setObjectGridAuthorization**(com.ibm.websphere.objectgrid.security.plugins.ObjectGridAuthorization ogAuthorization)

Sets the ObjectGridAuthorization for this ObjectGrid instance.

Passing null to this method removes a previously set ObjectGridAuthorization object from an earlier invocation of this method and indicates that this ObjectGrid is not associated with a ObjectGridAuthorization object.

This method should only be used when ObjectGrid security is enabled. If the ObjectGrid security is disabled, the provided ObjectGridAuthorization object will not be used.

A ObjectGridAuthorization plugin can be used to authorize access to the ObjectGrid and maps. Please refer to ObjectGridAuthorization for more details.

Note, to avoid an IllegalStateException, this method must be called prior to the initialize() method. Also, keep in mind that the getSession methods implicitly call the initialize() method if it has yet to be called by the application.

Parameters:
ogAuthorization - the ObjectGridAuthorization plugin
Throws:
IllegalStateException - if this method is called after the initialize() method is called.
Since:
WAS XD 6.1
See Also:
initialize(), ObjectGridAuthorization

---

setAccessByCreatorOnlyMode

void **setAccessByCreatorOnlyMode**(int accessByCreatorOnlyMode)

Set the "access by creator only" mode.

Enabling "access by creator only" mode ensures that only the user (represented by the Principals associated with it), who inserts the record into the map, can access (read, update, invalidate, and remove) the record.

The "access by creator only" mode can be disabled, or can complement the ObjectGrid authorization model, or it can supersede the ObjectGrid authorization model. The default value is disabled: SecurityConstants.ACCESS_BY_CREATOR_ONLY_DISABLED.

Parameters:
accessByCreatorOnlyMode - the access by creator mode.
Since:
WAS XD 6.1 FIX3
See Also:
SecurityConstants.ACCESS_BY_CREATOR_ONLY_DISABLED,
SecurityConstants.ACCESS_BY_CREATOR_ONLY_COMPLEMENT,
SecurityConstants.ACCESS_BY_CREATOR_ONLY_SUPERSEDE

---

getState

com.ibm.websphere.objectgrid.plugins.ObjectGridLifecycleListener.State **getState**()

Retrieve the current life cycle state of this ObjectGrid.

Returns:

the current state.

Since:

7.1.1

---

com.ibm.websphere.objectgrid

Class NoActiveTransactionException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.NoActiveTransactionException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class NoActiveTransactionException
extends ObjectGridException
```

An exception indicating there is no active transaction.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

NoActiveTransactionException() Constructs a new NoActiveTransactionException with `null` as its detail message.
NoActiveTransactionException(String message) Constructs a new NoActiveTransactionException with the specified detail message.
NoActiveTransactionException(String message, Throwable cause) Constructs a new NoActiveTransactionException with the specified detail message and cause.
NoActiveTransactionException(Throwable cause) Constructs a new NoActiveTransactionException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`()`

Constructs a new NoActiveTransactionException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

`ObjectGridException.initCause(Throwable)`

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`(String message)`

Constructs a new NoActiveTransactionException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`(String message,`
`Throwable cause)`

Constructs a new NoActiveTransactionException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this NoActiveTransactionException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`

NoActiveTransactionException

`public` **`NoActiveTransactionException`**`(Throwable cause)`

Constructs a new NoActiveTransactionException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for NoActiveTransactionExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class LockTimeoutException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException
          com.ibm.websphere.objectgrid.LockTimeoutException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

LockDeadlockException, LockInternalFailureException

```
public class LockTimeoutException
extends LockException
```

This exception is used by the lock manager to indicate that the maximum wait time for a lock has been exceeded. The timeout may or may not be the result of a deadlock. If it is a deadlock, the timeout is used to break the deadlock.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

LockTimeoutException() Constructs a new LockTimeoutException with `null` as its detail message.
LockTimeoutException(String message) Constructs a new LockTimeoutException with the specified detail message.
LockTimeoutException(String message, Throwable cause) Constructs a new LockTimeoutException with the specified detail message and cause.
LockTimeoutException(Throwable cause) Constructs a new LockTimeoutException with a specified cause.

Method Summary

void	**forceJavaCore**()
St	**getLockRequestQueueDetails**()

[ring](#)	Provides detailed information about the state of the lock queue when the lock timeout occurred.
[String](#)	**getMessage**() Returns the detail message string of this exception.
void	**setLockRequestQueueDetails**([String](#) string) Sets the details of the lock requests on the lock request queue at the time the lock timeout occurred.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockTimeoutException

public **LockTimeoutException**()

Constructs a new LockTimeoutException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

LockTimeoutException

public **LockTimeoutException**(String message)

Constructs a new LockTimeoutException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:
ObjectGridException.initCause(Throwable), getMessage()

LockTimeoutException

public **LockTimeoutException**(String message,
Throwable cause)

Constructs a new LockTimeoutException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LockTimeoutException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `getMessage()`

LockTimeoutException

`public` **`LockTimeoutException`**`(`Throwable` `cause)`

Constructs a new LockTimeoutException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LockTimeoutExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

Method Detail

getLockRequestQueueDetails

`public` `String` **`getLockRequestQueueDetails`**`()`

Provides detailed information about the state of the lock queue when the lock timeout occurred.

Returns:

the argument that was passed to the `setLockRequestQueueDetails(String)` method of this class or `null` if the `setLockRequestQueueDetails` method was not previously called for this object.

---

forceJavaCore

`public void` **`forceJavaCore`**`()`

---

setLockRequestQueueDetails

`public void` **`setLockRequestQueueDetails`**`(`String` `string)`

Sets the details of the lock requests on the lock request queue at the time the lock timeout occurred.

Parameters:

`string` - the details of lock requests on the lock request queue at the time the lock timeout occurred.

---

getMessage

public [String](#) **getMessage**()

Returns the detail message string of this exception. The returned String includes the request queue details as well as the message provided to the constructor.

Overrides:

[getMessage](#) in class [Throwable](#)

Returns:

the detail message string of this object instance

---

com.ibm.websphere.objectgrid

Class LockStrategy

java.lang.Object

  com.ibm.websphere.objectgrid.LockStrategy

```
public final class LockStrategy
extends Object
```

LockStrategy provides an enumerated type idiom for use on the `BackingMap.setLockStrategy(LockStrategy)` method. It determines whether or not a lock manager is needed for a BackingMap and if so, whether to use a optimistic or pessimistic locking strategy.

Since:

WAS XD 6.0, XC10

See Also:

`BackingMap.setLockStrategy(LockStrategy)`

Field Summary

static LockStrategy	**NONE** NONE indicates internal LockManager use is not needed since concurrency control is provided outside of ObjectGrid either by a persistence manager using objectgrid as a side cache, the application, or by a Loader plugin (for example, uses database locks to control concurrency).
static LockStrategy	**OPTIMISTIC** OPTIMISTIC is typically used for a map that does not have a Loader plugin, the map is read mostly, and locking is neither provided by a persistence manager using ObjectGrid as a side cache or by the application itself.
s	

static LockStrategy	**PESSIMISTIC** PESSIMISTIC is typically used for a map that does not have a Loader plugin and locking is neither provided by a persistence manager using ObjectGrid as a side cache, by a Loader plugin, or by the application itself.

Method Summary

String	**toString**() Returns a string representation of the `LockStrategy`.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

NONE

```
public static final LockStrategy NONE
```

NONE indicates internal LockManager use is not needed since concurrency control is provided outside of ObjectGrid either by a persistence manager using objectgrid as a side cache, the application, or by a Loader plugin (for example, uses database locks to control concurrency).

OPTIMISTIC

```
public static final LockStrategy OPTIMISTIC
```

OPTIMISTIC is typically used for a map that does not have a Loader plugin, the map is read mostly, and locking is neither provided by a persistence manager using ObjectGrid as a side cache or by the application itself. For this strategy, an exclusive lock is obtained on a map entry being inserted, updated, or removed at commit time. The lock ensures version information cannot be changed by another transaction while the transaction being committed is performing an optimistic version check.

PESSIMISTIC

```
public static final LockStrategy PESSIMISTIC
```

PESSIMISTIC is typically used for a map that does not have a Loader plugin and locking is neither provided by a persistence manager using ObjectGrid as a side cache, by a Loader plugin, or by the application itself. It is typically used when optimistic approach fails too often since there are update transactions that frequently collide on the same

map entry (e.g. not a read mostly map or large number of clients accessing a small map).

Method Detail

toString

public [String](#) **toString**()

Returns a string representation of the `LockStrategy`.

Overrides:

[toString](#) in class [Object](#)

Returns:

a string representation of the `LockStrategy`.

---

com.ibm.websphere.objectgrid

Class LockInternalFailureException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException
          com.ibm.websphere.objectgrid.LockTimeoutException
            com.ibm.websphere.objectgrid.LockInternalFailureException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **LockInternalFailureException**
extends LockTimeoutException

This exception is used by the lock manager to indicate it detected some internal programming error while processing a lock or unlock request.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

LockInternalFailureException() Constructs a new LockInternalFailureException with `null` as its detail message.
LockInternalFailureException(String message) Constructs a new LockInternalFailureException with the specified detail message.
LockInternalFailureException(String message, Throwable cause) Constructs a new LockInternalFailureException with the specified detail message and cause.
LockInternalFailureException(Throwable cause) Constructs a new LockInternalFailureException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.LockTimeoutException
forceJavaCore, getLockRequestQueueDetails, getMessage, setLockRequestQueueDetails

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockInternalFailureException

public **LockInternalFailureException**()

Constructs a new LockInternalFailureException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:
ObjectGridException.initCause(Throwable)

LockInternalFailureException

public **LockInternalFailureException**(String message)

Constructs a new LockInternalFailureException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:
`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.
See Also:
ObjectGridException.initCause(Throwable), LockTimeoutException.getMessage()

LockInternalFailureException

public **LockInternalFailureException**(String message,
Throwable cause)

Constructs a new LockInternalFailureException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LockInternalFailureException's detail message.

Parameters:
`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
See Also:
ObjectGridException.getCause(), LockTimeoutException.getMessage()

LockInternalFailureException

`public` **`LockInternalFailureException`**`(`[`Throwable`]` cause)`

Constructs a new LockInternalFailureException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LockInternalFailureExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class LockException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException

All Implemented Interfaces:

IObjectGridException, Serializable

Direct Known Subclasses:

LockTimeoutException

```
public class LockException
extends ObjectGridException
```

A general locking exception indicating something went wrong with locking operations.

Since:

WAS XD 6.0, XC10

See Also:

LockTimeoutException, Serialized Form

Constructor Summary

LockException() Constructs a new LockException with `null` as its detail message.
LockException(String message) Constructs a new LockException with the specified detail message.
LockException(String message, Throwable cause) Constructs a new LockException with the specified detail message and cause.
LockException(Throwable cause) Constructs a new LockException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockException

```
public LockException()
```

Constructs a new LockException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

LockException

```
public LockException(String message)
```

Constructs a new LockException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

LockException

```
public LockException(String message,
                     Throwable cause)
```

Constructs a new LockException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this LockException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

LockException

```
public LockException(Throwable cause)
```

Constructs a new LockException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for LockExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for

later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Class LockDeadlockException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.LockException
          com.ibm.websphere.objectgrid.LockTimeoutException
            com.ibm.websphere.objectgrid.LockDeadlockException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **LockDeadlockException**
extends LockTimeoutException

This exception is used by the lock manager to indicate that it detected a deadlock. It prevents the deadlock by throwing this exception. Typically, this deadlock is a result of the following scenario: one transaction owns a weaker lock as a result of getting a map entry, and then, at commit time, the transaction attempts to promote the weaker lock to a stronger lock in order to apply the changes to the data store. For example, two transactions try to promote from shared locks to exclusive locks but each transaction already owns a shared lock.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

LockDeadlockException() Constructs a new LockDeadlockException with `null` as its detail message.
LockDeadlockException(String message) Constructs a new LockDeadlockException with the specified detail message.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.LockTimeoutException
forceJavaCore, getLockRequestQueueDetails, getMessage, setLockRequestQueueDetails

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LockDeadlockException

public **LockDeadlockException**()

Constructs a new LockDeadlockException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

LockDeadlockException

public **LockDeadlockException**(String message)

Constructs a new LockDeadlockException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), LockTimeoutException.getMessage()

---

com.ibm.websphere.objectgrid

Class KeyNotFoundException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.KeyNotFoundException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **KeyNotFoundException**
extends ObjectGridException

Normally, record not found means a null is returned. However, sometimes on the explicit operation methods like update methods, we can figure that the record isn't there and then we throw this exception.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

KeyNotFoundException() Constructs a new KeyNotFoundException with null as its detail message.
KeyNotFoundException(String message) Constructs a new KeyNotFoundException with the specified detail message.
KeyNotFoundException(String message, Throwable cause) Constructs a new KeyNotFoundException with the specified detail message and cause.
KeyNotFoundException(Throwable cause) Constructs a new KeyNotFoundException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

KeyNotFoundException

public **KeyNotFoundException**()

Constructs a new KeyNotFoundException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

KeyNotFoundException

public **KeyNotFoundException**(String message)

Constructs a new KeyNotFoundException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

KeyNotFoundException

public **KeyNotFoundException**(String message,
Throwable cause)

Constructs a new KeyNotFoundException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this KeyNotFoundException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

KeyNotFoundException

public **KeyNotFoundException**(Throwable cause)

Constructs a new KeyNotFoundException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for KeyNotFoundExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that

the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Interface JavaMap

All Superinterfaces:

Map

```
public interface JavaMap
extends Map
```

This interface is a handle to a named Map. Maps should have homogeneous keys and values. An instance of this JavaMap can only be used by the thread that is currently associated with the Session that was used to get this JavaMap instance. Both Session and JavaMap objects are not allowed to be shared by multiple threads concurrently.

Users can get an instance of `JavaMap` from an instance of `ObjectMap` by calling `ObjectMap.getJavaMap()`. There are two main differences between `JavaMap` and `ObjectMap`:

- JavaMap extends java.util.Map. Therefore, users can cast an instance of `JavaMap` to java.util.Map if they want.
- The methods in `JavaMap` are defined to throw Exceptions similar to those defined on the `java.util.Map` interface that is `ObjectGridRuntimeException`, which is a subclass of `java.lang.RuntimeException` is used for error conditions. The methods in `ObjectMap` are defined to throw `ObjectGridException`s, which are checked exceptions.

The only methods that are supported from the `java.util.Map` interface are:

- containsKey(Object)
- get(Object)
- put(Object, Object)
- putAll(Map)
- remove(Object)
- clear()

All other methods on the `java.util.Map` interface will throw `java.lang.UnsupportedOperationException`.

Since:
WAS XD 6.0, XC10

See Also:
ObjectMap, ObjectMap.getJavaMap(), Map, BackingMap.setCopyMode(CopyMode, Class), BackingMap.setLockStrategy(LockStrategy)

Nested Class Summary

Nested classes/interfaces inherited from interface java.util.Map
Map.Entry<K,V>

Method Summary

void	**clear**() Clear all keys from the Map.
void	**clearCopyMode**() Resets the CopyMode back to the one in the BackingMap.
boolean	**containsKey**(Object key) Returns `true` if this map contains a mapping for the specified key.
boolean	**containsValue**(Object value) This method of the `java.util.Map` interface is not supported.
Set	**entrySet**() This method of the `java.util.Map` interface is not supported.
void	**flush**() Pushes the current set of changes for the `JavaMap` instance to the `Loader` without committing the changes.
Object	**get**(Object key) Retrieves the object from the cache at the given key.
List	**getAll**(List keyList) Gets a list of entries from the map.
List	**getAllForUpdate**(List keyList) Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries.
com.ibm.websphere.pro	

JectorMetadata	**getEntityMetadata**() Retrieve the metadata for the entity associated with this map.
Object	**getForUpdate**(Object key) Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry.
Object	**getIndex**(String name) Returns a reference to the named index that can be used with this Map.
String	**getName**() Returns the name of the JavaMap as defined by the configuration.
Object	**getNextKey**(long timeout) Retrieves a key off the map in first-in-first-out (FIFO) insert order.
void	**insert**(Object key, Object value) Performs an explicit insert of a given entry.
void	**invalidate**(Object key, boolean isGlobal) Invalidates an entry in the cache based on the key parameter.
void	**invalidateAll**(Collection keyList, boolean isGlobal) Invalidate a set of cache entries based on the Collection of keys provided.
boole	**isEmpty**() This method of the `java.util.Map` interface is not supported.

an	
Set	**keySet**() This method of the `java.util.Map` interface is not supported.
Object	**put**(Object key, Object value) Puts the Object value into the cache at location represented by key.
void	**putAll**(Map map) Puts each of the Object values into the cache at location represented by the corresponding key contained in the Map.
Object	**remove**(Object key) Removes the Object value from the cache represented by key.
void	**removeAll**(Collection keyList) Batch remove from the Map.
void	**setCopyMode**(CopyMode copyMode, Class valueInterface) Allows the CopyMode for the Map to be overridden on this map on this session only.
int	**setTimeToLive**(int ttl) Establishes the number of seconds that any given cache entry can live for, which is refered to as "time to live" or TTL.
int	**size**() This method of the `java.util.Map` interface is not supported.
void	**touch**(Object key) Updates the last access time in the BackingMap without retrieving the value to the JavaMap.
void	**update**(Object key, Object value) Performs an explicit update of a given entry.
Collection	**values**() This method of the `java.util.Map` interface is not supported.

Methods inherited from interface java.util.Map
equals, hashCode

Method Detail

getName

`String` **`getName`**`()`

Returns the name of the JavaMap as defined by the configuration.

Returns:
name of JavaMap

---

getForUpdate

`Object` **`getForUpdate`**`(Object key)`
`throws ObjectGridRuntimeException`

Same as `get(Object)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for this map entry. See `LockStrategy.PESSIMISTIC` for additional information. Whether or not a copy of the object is returned is determined by the `CopyMode` setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if the value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getForUpdate(Object)` for additional specification details.

Parameters:
`key` - The entry to fetch
Returns:
the value retrieved for update or `null`
Throws:
`IllegalArgumentException` - if key is `null`
`ObjectGridRuntimeException` - if an error occurs during processing
See Also:
`containsKey(Object)`, `get(Object)`, `CopyMode`, `LockStrategy.PESSIMISTIC`, `ObjectMap.getForUpdate(Object)`

---

getAll

`List` **`getAll`**`(List keyList)`
`throws ObjectGridRuntimeException`

Gets a list of entries from the map.

If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getAll(List)` for additional specification details.

Parameters:
`keyList` - A list of keys for identifying which entries to fetch
Returns:
a list of values
Throws:
`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing
See Also:
`get(Object)`, `ObjectMap.getAll(List)`

---

getAllForUpdate

```
List getAllForUpdate(List keyList)
                     throws ObjectGridRuntimeException
```

Same as the `getAll(List)` method except that if pessimistic lock strategy is used for this map, an upgradable lock mode is obtained for these map entries. See `LockStrategy.PESSIMISTIC` for additional information. If a key in the list cannot be found, a `null` value will be set at the appropriate position in the returned list.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

A return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getAllForUpdate(List)` for additional specification details.

Parameters:
`keyList` - A list of keys for identifying which entries to fetch
Returns:
a list of values
Throws:
`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing
See Also:
`getAll(List)`, `getForUpdate(Object)`, `LockStrategy.PESSIMISTIC`, `ObjectMap.getAllForUpdate(List)`

---

removeAll

```
void removeAll(Collection keyList)
               throws ObjectGridRuntimeException
```

Batch remove from the Map. If a key in the list cannot be found, it will be ignored.

See `ObjectMap.removeAll(Collection)` for additional specification details.

Parameters:
`keyList` - A list of keys for identifying which entries to remove
Throws:
`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

remove(Object), ObjectMap.removeAll(Collection)

---

invalidate

```
void invalidate(Object key,
                boolean isGlobal)
                throws ObjectGridRuntimeException
```

Invalidates an entry in the cache based on the key parameter.

If the key's value has changes pending in the JavaMap, it is the application's responsibility to flush these changes to the Loader before invalidation. If a flush is not performed prior to invoking the invalidate operation, all pending changes for this key will be removed from the JavaMap. If the key cannot be found in the map, it will be ignored.

The isGlobal parameter is used to indicate which cache level is used to invalidate the entries. If isGlobal is true, when the transaction is committed, the key is removed from the BackingMap also. If a subsequent get operation is performed, the BackingMap will be skipped and the Loader will be used to get the data. If isGlobal is false, the entry is only invalidated in the JavaMap (transactional cache). If a subsequent get operation is performed, the BackingMap can be used; and, if it's not in the BackingMap, the Loader will be used to get the data.

A typical use of isGlobal being false is when a large number of records are touched in a transaction and the application wants to evict records that are no longer used in the cache.

See `ObjectMap.invalidate(Object, boolean)` for additional specification details.

Parameters:

`key` - Object representing the key to be used for cache entry invalidation
`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the JavaMap (false).

Throws:

`IllegalArgumentException` - if key is `null`
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`ObjectMap.invalidate(Object, boolean)`

---

invalidateAll

```
void invalidateAll(Collection keyList,
                   boolean isGlobal)
                   throws ObjectGridRuntimeException
```

Invalidate a set of cache entries based on the Collection of keys provided. If a key in the collection cannot be found, it will be ignored.

See `ObjectMap.invalidateAll(Collection, boolean)` for additional specification details.

Parameters:

`keyList` - A Collection of keys representing the entries to be invalidated
`isGlobal` - Indicates whether to remove the entry from the BackingMap (true) or just the JavaMap (false).

Throws:

`IllegalArgumentException` - if keyList is null or contains a `null` element.
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`invalidate(Object, boolean)`, `ObjectMap.invalidateAll(Collection, boolean)`

---

setTimeToLive

```
int setTimeToLive(int ttl)
```

Establishes the number of seconds that any given cache entry can live for, which is refered to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this `JavaMap`, any previous value set by the `BackingMap.setTimeToLive(int)` method is overridden for this JavaMap. If this method is never called on the JavaMap, the TTL value from the BackingMap setting is used by default. If TTL is never set on the BackingMap, the cache entry can live "forever".

This method can only be used when the `TTLType` is set to `LAST_ACCESS_TIME` on the BackingMap. If this method is called on the JavaMap and the `TTLType` is something other than `LAST_ACCESS_TIME`, an IllegalStateException is thrown.

Parameters:

`ttl` - is the time-to-live value in seconds. The value must be >= 0. A value of 0 is used to indicate the cache entry can live "forever". Use of the constant `ObjectMap.TTL_FOREVER` is recommended when "forever" is desired.

Returns:

previous time-to-live value in seconds. The constant `ObjectMap.TTL_FOREVER` can be used to determine if the previous TTL was set to "forever".

Throws:

`IllegalArgumentException` - if seconds argument is < 0.

`IllegalStateException` - if `BackingMap.getTtlEvictorType()` returns anything other than `TTLType.LAST_ACCESS_TIME`.

See Also:

`BackingMap.setTimeToLive(int)`, `ObjectMap.setTimeToLive(int)`, `TTLType.LAST_ACCESS_TIME`

---

update

```
void update(Object key,
            Object value)
            throws ObjectGridRuntimeException
```

Performs an explicit update of a given entry.

A `get` operation is not required prior to invoking the `update` method (unlike the `put` method). Also, an `update` invocation will never insert a new record. If a the map's `LockStrategy` is `LockStrategy.OPTIMISTIC` this method will implicitly get the entry so as to have the version value of the object for when this method was invoked. Whether or not a copy of the object is made when transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If a key cannot be found in the map during commit, a TransactionException will be thrown.

See `ObjectMap.update(Object, Object)` for additional specification details.

Parameters:

`key` - Identifies the entry to be updated

`value` - The updated value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.

`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`insert(Object, Object)`, `put(Object, Object)`, `CopyMode`, `LockStrategy.OPTIMISTIC`,

ObjectMap.update(Object, Object)

---

insert

```
void insert(Object key,
            Object value)
            throws ObjectGridRuntimeException
```

Performs an explicit insert of a given entry.

The key must not exist before executing this method. Also, an `insert` invocation will never update an existing record. Whether or not a copy of the object is made when a transaction is committed is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode.

If the key is already in the map, a TransactionException will be thrown during commit.

See `ObjectMap.insert(Object, Object)` for additional specification details.

Parameters:

`key` - Identifies the entry to be inserted
`value` - The value for this entry

Throws:

`IllegalArgumentException` - if key is `null` or if the map does not allow `null` values and value is `null`.
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:

`put(Object, Object)`, `update(Object, Object)`, `CopyMode`, `ObjectMap.insert(Object, Object)`

---

getIndex

```
Object getIndex(String name)
                throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                       com.ibm.websphere.objectgrid.IndexNotReadyException,
                       UnsupportedOperationException
```

Returns a reference to the named index that can be used with this Map. This index cannot be shared between threads and works on the same rules as `Session`. The returned value should be cast to the right index interface such as `MapIndex`, `MapRangeIndex` or a custom index interface such as a geo spatial index.

Parameters:

`name` - The index name

Returns:

A reference to the index, it must be cast to the appropriate index interface.

Throws:

`IndexUndefinedException` - if the index is not defined on the BackingMap
`IndexNotReadyException` - if the index is a dynamic index and it is not ready
`UnsupportedOperationException` - if the map is a distributed map

Since:

WAS XD 6.0.1

---

flush

```
void flush()
          throws ObjectGridRuntimeException
```

Pushes the current set of changes for the `JavaMap` instance to the `Loader` without committing the changes. The changes are not propagated to the `BackingMap` either. This is useful for re-priming the `Loader`'s data without committing the current transaction and

starting over.

Throws:

ObjectGridRuntimeException - if an error occurs during processing

See Also:

Session.flush(), ObjectMap.flush()

---

size

`int` **`size`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:

size in interface Map

Returns:

the number of key-value mappings in this map.

Throws:

UnsupportedOperationException - indicating this method is not supported

---

isEmpty

`boolean` **`isEmpty`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:

isEmpty in interface Map

Returns:

`true` if this map contains no key-value mappings.

Throws:

UnsupportedOperationException - indicating this method is not supported

---

containsKey

`boolean` **`containsKey`**`(Object key)`

Returns `true` if this map contains a mapping for the specified key. ObjectGrid does not support null keys. If you configured the map to support `null` values, this method can be used to determine whether a key is contained in the map or not.

This API does not hold any locks when using pessimistic locking.

See ObjectMap.containsKey(Object) for additional specification details.

Specified by:

containsKey in interface Map

Parameters:

`key` - key whose presence in this map is to be tested.

Returns:

`true` if this map contains a mapping for the specified key.

Throws:

IllegalArgumentException - if `null` key parameter is passed in

ObjectGridRuntimeException - if an error occurs during processing

See Also:

ObjectMap.containsKey(Object)

---

containsValue

`boolean` **`containsValue`**`(Object value)`

This method of the `java.util.Map` interface is not supported.

Specified by:
`containsValue` in interface `Map`

Parameters:
`value` - value whose presence in this map is to be tested.

Returns:
`true` if this map maps one or more keys to the specified value.

Throws:
`UnsupportedOperationException` - indicating this method is not supported

---

get

`Object` **`get`**`(Object key)`

Retrieves the object from the cache at the given key.

Whether or not a copy of the object is returned is determined by the CopyMode setting for this map. See `CopyMode` for a description of each possible CopyMode. If the key cannot be found in the map, a `null` value will be returned. A `null` value is also returned if a value is `null` and this map allows `null` values. To distinguish the two, use the `containsKey` method.

The return value is a `SerializedValue` when using the `CopyMode.COPY_TO_BYTES_RAW` CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.get(Object)` for additional specification details.

Specified by:
`get` in interface `Map`

Parameters:
`key` - The entry to fetch

Returns:
the value, `null`, `SerializedValue` or `Tuple`

Throws:
`IllegalArgumentException` - if key is `null`
`ObjectGridRuntimeException` - if an error occurs during processing

See Also:
`containsKey(Object)`, `getForUpdate(Object)`, `CopyMode`, `ObjectMap.get(Object)`

---

put

`Object` **`put`**`(Object key,`
`Object value)`

Puts the Object value into the cache at location represented by key.

The values will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the `ObjectMap.setPutMode(PutMode)` property:

ObjectMap.PutMode.INSERTUPDATE	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
ObjectMap.PutMode.UPSERT	The values are put into the map using the specification of the ObjectMap.upsertAll(LinkedHashMap).

Whether or not a copy of the object is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

The return value is a `SerializedValue` when using the CopyMode.COPY_TO_BYTES_RAW CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See ObjectMap.put(Object, Object) for additional specification details.

Specified by:

put in interface Map

Parameters:

`key` - The entry to put into the map

`value` - The value to put into the map using the key

Returns:

If ObjectMap.PutMode.INSERTUPDATE is set, return the previous value in this transaction. If ObjectMap.PutMode.UPSERT is set, the return value is null.

Throws:

IllegalArgumentException - if key is `null`, or if the map does not allow `null` values and value is `null`

ObjectGridRuntimeException - if an error occurs during processing

See Also:

CopyMode, ObjectMap.put(Object, Object)

---

remove

Object **remove**(Object key)

Removes the Object value from the cache represented by key.

This removal will be pushed down to the BackingMap/Loader at commit time. If the key cannot be found in the map, a `null` value will be returned.

The return value is a `SerializedValue` when using the CopyMode.COPY_TO_BYTES_RAW CopyMode or `OutputFormat.RAW` OutputFormat with a ValueSerializerPlugin plug-in defined on the BackingMap. The SerializedValue allows access to the value in its serialized form, or its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See ObjectMap.remove(Object) for additional specification details.

Specified by:

remove in interface Map

Parameters:

`key` - The entry to remove

Returns:

the current value at invocation time

Throws:

IllegalArgumentException - if key is `null`

ObjectGridRuntimeException - if an error occurs during processing

See Also:

ObjectMap.remove(Object)

---

putAll

`void` **`putAll`**`(Map map)`

Puts each of the Object values into the cache at location represented by the corresponding key contained in the Map.

The value will be pushed down to the BackingMap/Loader at commit time and has two behaviors, which can be altered using the ObjectMap.setPutMode(PutMode) property:

ObjectMap.PutMode.INSERTUPDATE	(Deprecated) A put without a preceding get is an insert. For an entry in a map, a put following a get is always an update. However, if the entry is not in the map, a put following a get is an insert.
ObjectMap.PutMode.UPSERT	The value is put into the map using the specification of the ObjectMap.upsert(Object, Object).

Whether or not a copy of the objects contained in the map is made when transaction is committed is determined by the copy mode setting for this map. See `CopyMode` for a description of each possible copy mode.

An existing Map object will be passed in to use for obtaining the keys and values to be inserted or updated into the existing Map.

See ObjectMap.putAll(Map) for additional specification details.

Specified by:

putAll in interface Map

Parameters:

`map` - The key/values to be put into the map.

Throws:

IllegalArgumentException - if map is `null` or contains a `null` key or if `null` values are not allowed and map contains a `null` value.

ObjectGridRuntimeException - if an error occurs during processing

See Also:

put(Object, Object), ObjectMap.putAll(Map)

---

clear

`void` **`clear`**`()`

Clear all keys from the Map.

This method is an auto-commit call, so a session should not be explicitly begun or committed when calling clear on the Map.

Specified by:

clear in interface Map

Throws:

ObjectGridRuntimeException - if an error occurs during processing

Since:

keySet

`Set` **`keySet`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:
`keySet` in interface `Map`

Returns:
a set view of the keys contained in this map.
Throws:
`UnsupportedOperationException` - indicating this method is not supported

values

`Collection` **`values`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:
`values` in interface `Map`

Returns:
a collection view of the values contained in this map.
Throws:
`UnsupportedOperationException` - indicating this method is not supported

entrySet

`Set` **`entrySet`**`()`

This method of the `java.util.Map` interface is not supported.

Specified by:
`entrySet` in interface `Map`

Returns:
a set view of the mappings contained in this map.
Throws:
`UnsupportedOperationException` - indicating this method is not supported

touch

`void` **`touch`**`(``Object` `key)`

Updates the last access time in the BackingMap without retrieving the value to the JavaMap.

The last access time is updated during commit. If the key does not exist in the BackingMap, a TransactionException will be returned during commit processing.

See `ObjectMap.touch(Object)` for additional specification details.

Parameters:
`key` - key to be touched
Throws:

IllegalArgumentException - if key is `null`
ObjectGridRuntimeException - if an error occurs during processing

See Also:

ObjectMap.touch(Object)

---

setCopyMode

```
void setCopyMode(CopyMode copyMode,
                 Class valueInterface)
                 throws ObjectGridRuntimeException
```

Allows the CopyMode for the Map to be overridden on this map on this session only. This method allows an application to use an optimal CopyMode TRANSACTION by TRANSACTION as its needs dictate. The CopyMode cannot be changed during a transaction. There must be no active transaction when this method is called.

Parameters:

`copyMode` - must be one of the final static variables defined in `CopyMode`. See `CopyMode` class for an explanation of each mode and how the valueInterface is used for `CopyMode.COPY_ON_WRITE` .
`valueInterface` - the value interface Class object. Specify null in version 7.1 and later.

Throws:

IllegalArgumentException - if copyMode is `null` or COPY_ON_WRITE CopyMode is specified and the required value interface parameter is `null`
ObjectGridRuntimeException - if a transaction is active and this map has already been used in the transaction or an error occurs during processing

See Also:

BackingMap.setCopyMode(CopyMode, Class), CopyMode, ObjectMap.setCopyMode(CopyMode, Class)

---

clearCopyMode

```
void clearCopyMode()
                 throws ObjectGridRuntimeException
```

Resets the CopyMode back to the one in the BackingMap.

This method is used to reverse a previous setCopyMode method call for this JavaMap. This method can only be called when no transaction is active on the associated session.

Throws:

ObjectGridRuntimeException - if a transaction is active and this map has already been used in the transaction or an error occurs during processing

See Also:

setCopyMode(CopyMode, Class), ObjectMap.clearCopyMode()

---

getNextKey

```
Object getNextKey(long timeout)
```

Retrieves a key off the map in first-in-first-out (FIFO) insert order. The entry is locked by the session such that other calls to getNextKey will not return the same key. The key can be used to remove or manipulate the value although leaving the entry will result in the key remaining at the beginning of the queue. This order is optimized for performance and is not guaranteed especially across partitions or in highly concurrent environments.

The return value is a `SerializedKey` when OutputFormat.RAW is set for the keys. The default key output format for maps that are associated with a KeySerializerPlugin is OutputFormat.RAW. The SerializedKey allows access to the value in its serialized form, or

its native Java Object form.

The return value is a `Tuple` when an an EntityManager API entity is associated with the BackingMap.

See `ObjectMap.getNextKey(long)` for additional specification details.

Parameters:
`timeout` - The period of time to wait for an entry to become available on the queue.
Returns:
The next available key in the map.
See Also:
`ObjectMap.getNextKey(long)`

---

getEntityMetadata

`com.ibm.websphere.projector.md.EntityMetadata` **`getEntityMetadata`**`()`

Retrieve the metadata for the entity associated with this map.

Returns:
the EntityMetadata if an entity is associated with this map or null if there is no entity associated with this map.
Since:
WAS XD 6.1

---

com.ibm.websphere.objectgrid

Interface IObjectGridException

All Known Implementing Classes:

CacheEntryException, CannotGenerateCredentialException, ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException, com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException, ConnectException, ContinuousQueryAttributePathException, ContinuousQueryException, ContinuousQueryGetValueException, ContinuousQueryIncompatibleDuplicateException, DuplicateKeyException, ExpiredCredentialException, InvalidCredentialException, KeyNotFoundException, LoaderException, LockDeadlockException, LockException, LockInternalFailureException, LockTimeoutException, NoActiveTransactionException, ObjectGridConfigurationException, ObjectGridException, ObjectGridRuntimeException, ObjectGridSecurityException, ReadOnlyException, ReplicationVotedToRollbackTransactionException, SessionNotReentrantException, TransactionAffinityException, TransactionAlreadyActiveException, TransactionCallbackException, TransactionException, TransactionQuiesceException, TransactionTimeoutException, UnavailableServiceException, UndefinedMapException

---

```
public interface IObjectGridException
```

This interface is used to ensure JDK 1.4 Throwable chaining behavior for all exceptions thrown by ObjectGrid even when an earlier JDK is used (e.g. JDK 1.3.1).

Since:

WAS XD 6.0.1, XC10

Method Summary

Throwable	**getCause**() Provides JDK 1.4 Throwable.getCause() behavior.
Throwable	**initCause**(Throwable cause) Provides JDK 1.4 Throwable.initCause() behavior.

Method Detail

getCause

Throwable **getCause**()

Provides JDK 1.4 Throwable.getCause() behavior.

Returns the cause of this throwable or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this throwable to get thrown.)

This implementation returns the cause that was supplied via one of the constructors requiring a `Throwable`, or that was set after creation with the `initCause(Throwable)` method. While it is typically unnecessary to override this method, a subclass can override it to return a cause set by some other means. This is appropriate for a "legacy chained throwable" that predates the addition of chained exceptions to `Throwable`. Note that it is *not* necessary to override any of the `PrintStackTrace` methods, all of which invoke the `getCause` method to determine the cause of a throwable.

Returns:
the cause of this throwable or `null` if the cause is nonexistent or unknown.
See Also:
initCause(Throwable)

---

initCause

Throwable **initCause**(Throwable cause)
throws IllegalArgumentException,
IllegalStateException

Provides JDK 1.4 Throwable.initCause() behavior.

Initializes the *cause* of this throwable to the specified value. (The cause is the throwable that caused this throwable to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the throwable. If this throwable was created with `Throwable(Throwable)` or `Throwable(String,Throwable)`, this method cannot be called even once.

Parameters:
`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)
Returns:
a reference to this `Throwable` instance.
Throws:
IllegalArgumentException - if `cause` is this throwable. (A throwable cannot be its own cause.)
IllegalStateException - if this throwable was created with `Throwable(Throwable)` or `Throwable(String,Throwable)`, or this method has already been called on this throwable.

---

com.ibm.websphere.objectgrid

Class DuplicateKeyException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.DuplicateKeyException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class DuplicateKeyException
extends ObjectGridException
```

A DuplicateKeyException exception is thrown if a key cannot be inserted into a BackingMap because an object with the same key already exists.

Since:

WAS XD 6.0, XC10

See Also:

Serialized Form

Constructor Summary

DuplicateKeyException() Constructs a new DuplicateKeyException with `null` as its detail message.
DuplicateKeyException(String message) Constructs a new DuplicateKeyException with the specified detail message.
DuplicateKeyException(String message, Throwable cause) Constructs a new DuplicateKeyException with the specified detail message and cause.
DuplicateKeyException(Throwable cause) Constructs a new DuplicateKeyException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

DuplicateKeyException

```
public DuplicateKeyException()
```

Constructs a new DuplicateKeyException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

[ObjectGridException.initCause(Throwable)]

DuplicateKeyException

```
public DuplicateKeyException(String message)
```

Constructs a new DuplicateKeyException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

[ObjectGridException.initCause(Throwable)], [Throwable.getMessage()]

DuplicateKeyException

```
public DuplicateKeyException(String message,
                             Throwable cause)
```

Constructs a new DuplicateKeyException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this DuplicateKeyException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

[ObjectGridException.getCause()], [Throwable.getMessage()]

DuplicateKeyException

```
public DuplicateKeyException(Throwable cause)
```

Constructs a new DuplicateKeyException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for DuplicateKeyExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

ObjectGridException.getCause()

---

com.ibm.websphere.objectgrid

Class CopyMode

java.lang.Object

  com.ibm.websphere.objectgrid.CopyMode

```
public final class CopyMode
extends Object
```

This class is used to define the "copy" mode when the `setCopyMode` method of the `BackingMap` interface is used. The application is expected to pass one of the final static variables that are defined in this class to the `setCopyMode` method.

Since:

WAS XD 6.0, XC10

See Also:

`BackingMap.setCopyMode(CopyMode, Class)`, `ObjectTransformer.copyValue(Object)`

Field Summary

static CopyMode	**COPY_ON_READ** The COPY_ON_READ mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when a transaction is committed.
static CopyMode	**COPY_ON_READ_AND_COMMIT** The COPY_ON_READ_AND_COMMIT mode is the default mode.
static CopyMode	**COPY_ON_WRITE** The COPY_ON_WRITE mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when `ObjectMap.get` is called for the first time by a transaction for a given key.

[Mode](#)	
static [CopyMode](#)	**COPY_TO_BYTES** The COPY_TO_BYTES mode is similar to the COPY_ON_READ_AND_COMMIT mode in that it ensures that an application never has a reference to the value object that is in the BackingMap.
static [CopyMode](#)	**COPY_TO_BYTES_RAW** When set, all ObjectMap APIs that return a `SerializedValue` rather than the original Java Object, allowing access to the serialized form of the data, preventing inflation of object into Java Object form.
static [CopyMode](#)	**NO_COPY** The NO_COPY mode allows an application to promise that it will never modify a value object obtained using an ObjectMap.get method in exchange for performance improvements.

Method Summary

boolean	**isBytes**() Is the copy mode one of the copy modes that indicate copy to bytes?
[String](#)	**toString**() Returns a string representation of the `CopyMode`.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

COPY_ON_READ_AND_COMMIT

`public static final CopyMode` **`COPY_ON_READ_AND_COMMIT`**

The COPY_ON_READ_AND_COMMIT mode is the default mode. This mode ensures that an application never has a reference to the value object that is in the BackingMap, and instead the application is always working with a copy of the value that is in the BackingMap. The copy ensures the application can never inadvertently corrupt the data that is cached in the BackingMap. When an application transaction calls an `ObjectMap.get` method for a given key, and it is the first access of the ObjectMap entry for that key, a copy of the value is returned. When the transaction is committed, any changes the application committed are copied to the BackingMap to ensure that the application does not have reference to the committed value in the BackingMap.

COPY_ON_READ

`public static final CopyMode` **`COPY_ON_READ`**

The COPY_ON_READ mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when a transaction is committed. To preserve integrity of BackingMap data, the application promises to destroy every reference it has to an entry once the transaction is committed. This mode results in a `ObjectMap.get` method returning a copy of the value rather than a reference to the value to ensure that changes made by the application to the value does not affect the BackingMap value until the transaction is committed. However, when the transaction does commit, a copy of changes is not made. Instead, the reference to the copy that was returned by ObjectMap.get is stored in the BackingMap. This is the reason the application must agree to destroy all map entry references once the transaction is committed. If application fails to keep its promise, the application could cause the data cached in BackingMap to become corrupted. If an application is using this mode and it is having problems, then switch to the COPY_ON_READ_AND_COMMIT mode to see if the problem still exists. If the problem goes away, then more than likely the application is failing to destroy all of its references after the transaction has committed.

COPY_ON_WRITE

`public static final CopyMode` **`COPY_ON_WRITE`**

The COPY_ON_WRITE mode improves performance over the COPY_ON_READ_AND_COMMIT mode by eliminating the copy that occurs when `ObjectMap.get` is called for the first time by a transaction for a given key. Instead, the ObjectMap.get method returns a proxy to the value rather than a direct reference to the value object itself. The proxy ensures that a copy of the value is not made unless the application calls a set method on the value interface that is passed on the `BackingMap.setCopyMode(CopyMode, Class)` method. Thus, the proxy provides a "copy on write" implementation. When a transaction commits, the `BackingMap` examines the proxy to determine if any copy was made as a result of a set method being called. If a copy was made, then the reference to that copy is stored in the BackingMap. The big advantage of this mode is a value is never copied on read or at commit when the transaction never calls a set method to mutate the value.

See Also:

`BackingMap.setCopyMode(CopyMode, Class)`

NO_COPY

`public static final CopyMode` **`NO_COPY`**

The NO_COPY mode allows an application to promise that it will never modify a value

object obtained using an ObjectMap.get method in exchange for performance improvements. If this mode is used, no copy of the value is ever made. If the application breaks its promise and does modify values, then data in the BackingMap will be corrupted. This mode is primarily useful for read only maps where data is never modified by the application. If the application is using this mode and it is having problems, then switch to COPY_ON_READ_AND_COMMIT mode to see if the problem still exists. If the problem goes away, then more than likely the application is not keeping its promise and is modifying the value returned by ObjectMap.get method (either during transaction or after transaction has committed).

COPY_TO_BYTES

`public static final CopyMode COPY_TO_BYTES`

The COPY_TO_BYTES mode is similar to the COPY_ON_READ_AND_COMMIT mode in that it ensures that an application never has a reference to the value object that is in the BackingMap. The value that the application works with is a newly inflated version of the serialized version that is in the BackingMap. The copy ensures the application can never inadvertently corrupt the data that is cached in the BackingMap since a byte form of the value is what is stored in the BackingMap instead of the Object form.

A copy of the value is returned when an application transaction calls an `ObjectMap.get` method for a given key, and it is the first time that the ObjectMap entry is accessed for that key. When the transaction is committed, any changes the application committed are copied to bytes in the BackingMap to ensure that the application does not have reference to the committed value in the BackingMap.

Since:
7.0

COPY_TO_BYTES_RAW

`public static final CopyMode COPY_TO_BYTES_RAW`

When set, all ObjectMap APIs that return a `SerializedValue` rather than the original Java Object, allowing access to the serialized form of the data, preventing inflation of object into Java Object form.

Since:
7.1.1
See Also:
`ValueDataSerializer`

Method Detail

toString

`public String toString()`

Returns a string representation of the `CopyMode`.

Overrides:
`toString` in class `Object`

Returns:
a string representation of the `CopyMode`.

---

isBytes

`public boolean isBytes()`

Is the copy mode one of the copy modes that indicate copy to bytes?

Returns:

boolean indicating if copy mode is one of COPY_TO_BYTES or COPY_TO_BYTES_RAW.

---

com.ibm.websphere.objectgrid

Class ConnectException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.ConnectException

All Implemented Interfaces:

IObjectGridException, Serializable

---

public class **ConnectException**
extends ObjectGridException

This exception is used to indicate that the client was unable to connect to the server

Since:

WAS XD 6.0.1, XC10

See Also:

Serialized Form

Field Summary

static int	**BAD_CONFIGURATION** This failure code indicates the provided configuration was corrupt.
static int	**CONNECTION_REFUSED** This failure code indicates that the server may not be available.
static int	**FAILED_SECURITY** This failure code indicates the a failure to authenticate.
s	

static int	**SERVER_DEFINITION_NOT_FOUND** This failure code indicates the definition of cluster cannot be accessed.
static int	**UNKNOWN** This failure code indicates the reason for the connect failure is unknown.

Constructor Summary

ConnectException() Constructs a new ConnectException with `null` as its detail message.
ConnectException(String message) Constructs a new ConnectException with the specified detail message.
ConnectException(String message, int failureCode) Constructs a new ConnectException with the specified detail message.
ConnectException(String message, Throwable cause) Constructs a new ConnectException with the specified detail message and cause.
ConnectException(String message, Throwable cause, int failureCode) Constructs a new ConnectException with the specified detail message and cause.
ConnectException(Throwable cause) Constructs a new ConnectException with a specified cause.

Method Summary

int	**getFailureCode**() Returns the failure code that was set by one of the constructors that accepts a failure code, or `UNKNOWN` if one of the other constructors was called.

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException

getCause, initCause

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

BAD_CONFIGURATION

public static final int **BAD_CONFIGURATION**

This failure code indicates the provided configuration was corrupt.

See Also:
Constant Field Values

UNKNOWN

`public static final int` **`UNKNOWN`**

This failure code indicates the reason for the connect failure is unknown.

See Also:
Constant Field Values

FAILED_SECURITY

`public static final int` **`FAILED_SECURITY`**

This failure code indicates the a failure to authenticate.

See Also:
Constant Field Values

CONNECTION_REFUSED

`public static final int` **`CONNECTION_REFUSED`**

This failure code indicates that the server may not be available.

See Also:
Constant Field Values

SERVER_DEFINITION_NOT_FOUND

`public static final int` **`SERVER_DEFINITION_NOT_FOUND`**

This failure code indicates the definition of cluster cannot be accessed.

See Also:
Constant Field Values

Constructor Detail

ConnectException

`public` **`ConnectException`**`()`

Constructs a new ConnectException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method. In addition the failure code is initialized to `UNKNOWN`.

See Also:
`ObjectGridException.initCause(Throwable)`, `UNKNOWN`

ConnectException

`public` **`ConnectException`**`(`String message`)`

Constructs a new ConnectException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method. In addition the failure code is initialized to `UNKNOWN`.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`, `UNKNOWN`

ConnectException

```
public ConnectException(String message,
                        int failureCode)
```

Constructs a new ConnectException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

`failureCode` - the failure code which should be one of the constants of this exception class.

See Also:

`ObjectGridException.initCause(Throwable)`, `Throwable.getMessage()`, `getFailureCode()`

ConnectException

```
public ConnectException(Throwable cause)
```

Constructs a new ConnectException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ConnectExceptions that are little more than wrappers for other throwables. The failure code is initialized to `UNKNOWN`.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`, `UNKNOWN`

ConnectException

```
public ConnectException(String message,
                        Throwable cause)
```

Constructs a new ConnectException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ConnectException's detail message. The failure code is initialized to `UNKNOWN`.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).

`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

ObjectGridException.getCause(), Throwable.getMessage()

ConnectException

```
public ConnectException(String message,
                        Throwable cause,
                        int failureCode)
```

Constructs a new ConnectException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ConnectException's detail message. The failure code is initialized to `UNKNOWN`.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).
`failureCode` - the failure code which should be one of the constants of this exception class.

See Also:

ObjectGridException.getCause(), Throwable.getMessage(), getFailureCode()

Method Detail

getFailureCode

```
public int getFailureCode()
```

Returns the failure code that was set by one of the constructors that accepts a failure code, or `UNKNOWN` if one of the other constructors was called.

Returns:

the failure code. One of the constants of this exception class.

---

com.ibm.websphere.objectgrid

Class ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      com.ibm.websphere.objectgrid.ObjectGridException
        com.ibm.websphere.objectgrid.plugins.TransactionCallbackException
          com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException
            com.ibm.websphere.objectgrid.ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

All Implemented Interfaces:

IObjectGridException, Serializable

```
public class ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException
extends com.ibm.websphere.objectgrid.ClientServerTransactionCallbackException
```

This exception is thrown when a method call to the Client/Server TransactionCallback detects the user is attempting to perform a write against multiple maps in different Map Sets, Partition Sets or Replication groups. This is not allowed.

Since:

WAS XD 6.0.1, XC10

See Also:

TransactionCallback, Serialized Form

Constructor Summary

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException() Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with `null` as its detail message.
ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException(String message) Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message.
ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException(String message, Throwable cause) Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message and cause.
ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException(Throwable cause) Constructs a new

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with a specified cause.

Method Summary

Methods inherited from class com.ibm.websphere.objectgrid.ObjectGridException
getCause, initCause

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, getStackTrace, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

public **ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException**()

Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with `null` as its detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

See Also:

ObjectGridException.initCause(Throwable)

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

public **ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException**(String message)

Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message. The cause is not initialized, and may subsequently be initialized by a call to the `initCause` method.

Parameters:

`message` - the detail message. The detail message is saved for later retrieval by the `getMessage` method.

See Also:

ObjectGridException.initCause(Throwable), Throwable.getMessage()

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

public **ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException**(String message, Throwable cause)

Constructs a new

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with the specified detail message and cause.

Note that the detail message associated with `cause` is *not* automatically incorporated in this ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException's detail message.

Parameters:

`message` - the detail message (which is saved for later retrieval by the `getMessage` method).
`cause` - the cause (which is saved for later retrieval by the `getCause` method). (A`null` value is permitted, and indicates that the cause is nonexistent or unknown).

See Also:

`ObjectGridException.getCause()`, `Throwable.getMessage()`

ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException

`public` **`ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException`**`(Throwable cause)`

Constructs a new ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackException with a specified cause. The cause and a detail message of `(cause==null ? null : cause.toString())` is used (which typically contains the class and detail message of `cause`). This constructor is useful for ClientServerMultipleReplicationGroupMemberWriteTransactionCallbackExceptions that are little more than wrappers for other throwables.

Parameters:

`cause` - is the exception that caused this exception to be thrown, which is saved for later retrieval by the `getCause()` method. A `null` value is permitted and indicates that the cause is nonexistent or is unknown.

See Also:

`ObjectGridException.getCause()`

---

com.ibm.websphere.objectgrid

Interface ClientReplicableMap

All Known Subinterfaces:

BackingMap

Deprecated. *The client replicated map function is deprecated in version 8.6. Use the `ContinuousQueryManager` function.*

```
public interface ClientReplicableMap
```

This interface represents a replicable client map. A replicable client map can be a continuous replica or a snapshot replica of the server map.

If the client is a continuous replica of the server map, the data in the server will be replicated to the client continuously in an asynchronous manner.

If the client is a snapshot replica of the server map, a snapshot on the data in the server will be taken and the snapshot will be replicated to the client in an asynchronous manner. A snapshot replication is a one-time replication.

A ReplicationMapListener can be used to listen for the data changes as well as the replication lifecycle events.

Since:
WAS XD 6.1, XC10

See Also:
ReplicationMapListener

Nested Class Summary

static class	**ClientReplicableMap.Mode** **Deprecated.** Client Replication mode

Field Summary

static Clie	

[IClientReplicableMap.Mode](#)	**CONTINUOUS_REPLICATION** **Deprecated.** Full replication mode.
[IClientReplicableMap.Mode](#)	**NONE** **Deprecated.** No replication mode, aka normal mode.
[IClientReplicableMap.Mode](#)	**SNAPSHOT_REPLICATION** **Deprecated.** Snapshot replication mode.

[o d e](#)	

Method Summary

void	**disableClientReplication**() **Deprecated.** Disables the replication for this client.
void	**enableClientReplication**(ClientReplicableMap.Mode mode, int[] partitions, ReplicationMapListener listener) **Deprecated.** Make the client map a replica of the server side map.
ClientReplicableMap.Mode	**getReplicationMode**() **Deprecated.** Returns the current replication mode

Field Detail

NONE

static final ClientReplicableMap.Mode **NONE**

Deprecated.
No replication mode, aka normal mode.

CONTINUOUS_REPLICATION

static final ClientReplicableMap.Mode **CONTINUOUS_REPLICATION**

Deprecated.
Full replication mode. Data in the server map will be replicated to the client continuously.

SNAPSHOT_REPLICATION

static final ClientReplicableMap.Mode **SNAPSHOT_REPLICATION**

Deprecated.

Snapshot replication mode. A snapshot on the data in the server will be taken and the snapshot will be replicated to the client. A snapshot replication is a one-time replication.

Method Detail

enableClientReplication

```
void enableClientReplication(ClientReplicableMap.Mode mode,
                             int[] partitions,
                             ReplicationMapListener listener)
                             throws ObjectGridException
```

Deprecated.
Make the client map a replica of the server side map.

When security is enabled, this method requires a `ServerMapPermission` with action "replicate". Refer to `ServerMapPermission` for more permission details.

Required Client Permission: `ServerMapPermission.REPLICATE`

Parameters:

`mode` - The replication mode.
`partitions` - The array of partition IDs represent which partitions the data should be replicated from. If the value is null or an empty array, it indicates the data should be replicated from all partitions.
`listener` - a listener to receive client replication events

Throws:

`IllegalArgumentException` - if mode is not `CONTINUOUS_REPLICATION` or `SNAPSHOT_REPLICATION` or the map isn't currently in the mode specified or is not in `NONE` mode
`IllegalStateException` - if this method is invoked on a map other than a client map
`ObjectGridException` - if an error occurs during processing this request

See Also:

`ReplicationMapListener`, `CONTINUOUS_REPLICATION`, `SNAPSHOT_REPLICATION`, `getReplicationMode()`, `BackingMap.CLIENT`

---

getReplicationMode

```
ClientReplicableMap.Mode getReplicationMode()
```

Deprecated.
Returns the current replication mode

Returns:

the replication mode

See Also:

`NONE`, `CONTINUOUS_REPLICATION`, `SNAPSHOT_REPLICATION`

---

disableClientReplication

```
void disableClientReplication()
                            throws ObjectGridException
```

Deprecated.
Disables the replication for this client. If it is not in a replication mode, this method will be a no-op.

When security is enabled, this method requires a `ServerMapPermission` with action "replicate". Refer to `ServerMapPermission` for more permission details.

Throws:

IllegalStateException - if this method is invoked on a map other than a client map
ObjectGridException - if an error occurs during processing this request

See Also:

BackingMap.CLIENT

---

com.ibm.websphere.objectgrid

Class ClientReplicableMap.Mode

java.lang.Object

com.ibm.websphere.objectgrid.ClientReplicableMap.Mode

Enclosing interface:

ClientReplicableMap

```
public static final class ClientReplicableMap.Mode
extends Object
```

Client Replication mode

Method Summary

String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Method Detail

toString

```
public String toString()
```

Overrides:

toString in class Object

com.ibm.websphere.objectgrid

Interface ClientClusterContext

public interface **ClientClusterContext**

This interface is a context to represent which cluster/domain the client connected to using one of the `ObjectGridManager.connect` methods. An instance of this interface is used to retrieve client `ObjectGrid` instances and for performing admin operations against an ObjectGrid cluster/domain or its servers.

Since:
WAS XD 6.0.1, XC10

See Also:
ObjectGridManager

Method Summary

ClientProperties	**getClientProperties**(String objectGridName) Retrieve the ClientProperties object for this ClientClusterContext for the specified ObjectGrid name.
String	**getClusterName**() Gets the name of the domain to which the client is connected
void	**setClientProperties**(String objectGridName, URL url) Sets the ClientProperties properties for the selected ObjectGrid using the specified client properies file.

Method Detail

getClusterName

String **getClusterName**()

Gets the name of the domain to which the client is connected

Returns:
the name of the domain this context is connected to

---

getClientProperties

`ClientProperties` **`getClientProperties`**`(String objectGridName)`

Retrieve the ClientProperties object for this ClientClusterContext for the specified ObjectGrid name. A ClientProperties is scoped to this ClientClusterContext and a single ObjectGrid.

Parameters:
`objectGridName` - the name of ObjectGrid
Returns:
the ClientProperties instance for this ObjectGrid.
Since:
WAS XD 6.1.0.3

---

setClientProperties

```
void setClientProperties(String objectGridName,
                         URL url)
```

Sets the ClientProperties properties for the selected ObjectGrid using the specified client properies file.

To further adjust the client properties, call the `getClientProperties(String)` method.

Parameters:
`objectGridName` - the name of ObjectGrid to apply the ClientProperties to.
`url` - the URL where the client properties file can be located.
Since:
WAS XD 6.1.0.3

---

com.ibm.websphere.objectgrid

Interface CatalogDomainManager

public interface **CatalogDomainManager**

Provides access to catalog domain configuration information for the current environment.

When running in a WebSphere Application Server profile augmented with WebSphere eXtreme Scale, the CatalogDomainManager returns the catalog service domain configuration information that is configured in the administration console.

Since:
8.5, XC10

See Also:
ObjectGridManager.getCatalogDomainManager()

Method Summary

CatalogDomainInfo	**getDefaultDomainInfo**() Retrieve the default, configured CatalogDomainInfo.
CatalogDomainInfo	**getDomainInfo**(String domainId) Retrieve the specified CatalogDomainInfo for the specified domainId.
Coll	

Collection<CatalogDomainInfo>	**getDomainInfos**() Retrieve all configured CatalogDomainInfo objects.

Method Detail

getDefaultDomainInfo

CatalogDomainInfo **getDefaultDomainInfo**()

Retrieve the default, configured CatalogDomainInfo.

Returns:
the default CatalogDomainInfo, or null if not available.

---

getDomainInfo

CatalogDomainInfo **getDomainInfo**(String domainId)

Retrieve the specified CatalogDomainInfo for the specified domainId.

Parameters:
`domainId` - the domain identifier.
Returns:
the CatalogDomainInfo if found, or null.

---

getDomainInfos

Collection<CatalogDomainInfo> **getDomainInfos**()

Retrieve all configured CatalogDomainInfo objects.

Returns:
a collection of CatalogDomainInfo

---

com.ibm.websphere.objectgrid

Interface CatalogDomainInfo

public interface **CatalogDomainInfo**

Identifies the configuration attributes of a catalog service domain.

Since:
8.5, XC10

See Also:
CatalogDomainManager

Method Summary

String	**getClientCatalogServerEndpoints**() Retrieve the catalog server endpoints used to connect a client to the remote catalog service domain.
ClientSecurityConfiguration	**getClientSecurityConfiguration**() Retrieve the ClientSecurityConfiguration for the domain.
String	**getDomainId**() Retrieve the identifier of the domain as specified by the catalog service domain.

Method Detail

getDomainId

String **getDomainId**()

Retrieve the identifier of the domain as specified by the catalog service domain.

Note: This is different than the name of the domain which is specified when starting the catalog services.

Returns:
the identifier of the domain.

---

getClientCatalogServerEndpoints

String **getClientCatalogServerEndpoints**()

Retrieve the catalog server endpoints used to connect a client to the remote catalog service domain.

The catalog service endpoints are used with the ObjectGridManager.connect(String, com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration, java.net.URL) method to connect to a catalog service domain.

Returns:
the catalog service endpoints.

---

getClientSecurityConfiguration

ClientSecurityConfiguration **getClientSecurityConfiguration**()

Retrieve the ClientSecurityConfiguration for the domain.

The ClientSecurityConfiguration are used with the ObjectGridManager.connect(String, com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration, java.net.URL) method to connect to a catalog service domain.

Returns:
the ClientSecurityConfiguration or null if security is not configured.

---

com.ibm.websphere.objectgrid

Interface BackingMap

All Superinterfaces:

ClientReplicableMap

```
public interface BackingMap
extends ClientReplicableMap
```

This is the public interface to the BackingMap. It is returned when a new Map is defined on the ObjectGrid. It allows the Map to be customized with various plug-ins or by setting properties. The defaults are:

- No external `Evictor`, but an internal time-based evictor is provided by default
- No `Loader`
- No `EventListeners`
- No `MapIndexPlugins`
- An internal `ObjectTransformer`
- An internal `OptimisticCallback`
- Key is not copied
- A value `CopyMode` of `CopyMode.COPY_ON_READ_AND_COMMIT`
- A `LockStrategy` of `LockStrategy.OPTIMISTIC`
- A default lock timeout
- `null` values are supported
- A default number of buckets
- A default number of lock buckets
- Synchronous preload
- Read/write map by default
- A TimeToLive of 0 (indicating unlimited time)
- A TtlEvictor type of `TTLType.NONE`
- Write-behind updates is disabled
- Time-based database updates are disabled
- Eviction triggers are not set

Since:

WAS XD 6.0, XC10

Nested Class Summary

Nested classes/interfaces inherited from interface com.ibm.websphere.objectgrid.ClientReplicableMap
ClientReplicableMap.Mode

Field Summary

static	**CLIENT**

static int	Constant used to indicate this map is a client to a server map
static int	**DEFAULT_LOCK_TIMEOUT** Default lock timeout used if `setLockTimeout(int)` is not invoked.
static int	**DEFAULT_NUMBER_OF_BUCKETS** **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the [isNearCacheEnabled()] flag to disable the near cache in the ObjectGrid configuration XML file.*
static int	**DEFAULT_NUMBER_OF_LOCK_BUCKETS** Default number of lock buckets used if `setNumberOfLockBuckets(int)` is not invoked.
static String	**EVICTIONTRIGGER_MEMORY_USAGE_THRESHOLD** The eviction trigger string constant to enable memory based eviction using memory usage threshold provided by the `java.lang.management.MemoryPoolMXBean`.
static int	**LOCAL** Constant used to indicate this map is not a distributed map.
static int	**SERVER** Constant used to indicate this map is a server map.

Fields inherited from interface com.ibm.websphere.objectgrid.ClientReplicableMap
CONTINUOUS_REPLICATION, NONE, SNAPSHOT_REPLICATION

Method Summary

void	**addMapEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener eventListener) Adds an EventListener to this BackingMap.
void	**addMapEventListener**(MapEventListener eventListener) **Deprecated.** *This method is deprecated in version 7.1.1, use the addMapEventListener(EventListener) method.*
void	**addMapIndexPlugin**(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index) Adds an MapIndexPlugin to this Map.
void	**createDynamicIndex**(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index, com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicIndexCallback) Creates a dynamic index on the BackingMap.
void	**createDynamicIndex**(String name, boolean isRangeIndex, String attributeName, com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicIndexCallback) Creates a dynamic index on the BackingMap.
boolean	**getCopyKey**() Gets whether keys are copied for this BackingMap.
CopyMode	**getCopyMode**() Gets the CopyMode being used by this BackingMap.
com.ibm.websphere.projector.	**getEntityMetadata**() Retreive the metadata for the entity associated with this backing map.

md.EntityMetadata	
String	**getEvictionTriggers**() Returns the types of additional eviction triggers.
Evictor	**getEvictor**() Gets the `Evictor` being used by this BackingMap.
com.ibm.websphere.objectgrid.OutputFormat	**getKeyOutputFormat**() Retrieves the data format for all data access APIs that return cache keys.

com.ibm.websphere.objectgrid.plugins.Loader	**getLoader**() Gets the Loader being used by this BackingMap.
LockStrategy	**getLockStrategy**() Gets the LockStrategy object being used by this BackingMap.
int	**getLockTimeout**() Gets the lock timeout value used by the lock manager for this BackingMap.
List	**getMapEventListeners**() Gets the current list of EventListenerS.
List	**getMapIndexPlugins**() Returns the current list of MapIndexPlugin objects for this BackingMap.
String	**getMapSetName**() Retrieves the name of the MapSet that this BackingMap is currently associated with.

[ng](#)	
int	**getMapType**() Returns the type of BackingMap.
[String](#)	**getName**() Gets the name of the `BackingMap`.
boolean	**getNullValuesSupported**() Gets whether this `BackingMap` supports `null` values or not.
int	**getNumberOfBuckets**() **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the [isNearCacheEnabled()](#) flag to disable the near cache in the ObjectGrid configuration XML file.*
int	**getNumberOfLockBuckets**() Gets the number of lock buckets defined for the hash map used by lock manager for this backing map.
[ObjectGrid](#)	**getObjectGrid**() Gets the `ObjectGrid` that owns this `BackingMap`.
com.ibm.websphere.objectgrid.plug	**getObjectTransformer**() Gets the `ObjectTransformer` object being used by this `BackingMap` and/or `Loader`.

ins.ObjectTransformer	
com.ibm.websphere.objectgrid.plugins.OptimisticCallback	**getOptimisticCallback**() Gets the OptimisticCallback being used by this BackingMap and/or Loader or null if the LockStrategy is not optimistic.

int	**getPartitionId**() Gets the partition identifier being used by this BackingMap.
PartitionManager	**getPartitionManager**() Allows access to the PartitionManager that is defined for this BackingMap.
boolean	**getPreLoadMode**() Returns whether this BackingMap will be asynchronously preloaded or not if a Loader is set.
boolean	**getReadOnly**() Retrieves the map type.
com.ibm.websphere.objectgrid.plugins.	**getSerializerAccessor**() Retrieve the SerializerAccessor for this map.

io.SerializerAccessor	
com.ibm.websphere.objectgrid.plugins.BackingMapLifecycl	**getState**() Retrieve the current life cycle state of this map.

eListener.State	
com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig	**getTimeBasedDBUpdateConfig**() Get the time-based database update configuration object.
int	**getTimeToLive**() Gets the number of seconds for an entry to live.
TTL	**getTtlEvictorType**()

Type	
	Gets how expiration time of a `BackingMap` entry is computed.
com.ibm.websphere.objectgrid.OutputFormat	**getValueOutputFormat**() Retrieves the data output format for all data access APIs that return cache values.
String	**getWriteBehind**() Get the write-behind parameter.
boolean	**isNearCacheEnabled**() If true, the client near cache is enabled for supported configurations.
boolean	**isNearCacheInvalidationEnabled**() If true, clients with local caches are automatically invalidated when the data grid map is updated.
boole	**isNearCacheLastAccessTTLSyncEnabled**() If true, clients automatically send time-to-live access information to the remote data grid when accessed and the TTLType.LAST_ACCESS_TIME TTL evictor type is configured.

an	
void	**removeDynamicIndex**(String name) Removes a dynamic index on the BackingMap.
void	**removeMapEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener eventListener) Removes an EventListener from this BackingMap.
void	**removeMapEventListener**(MapEventListener eventListener) Provided for compatibility with old releases, use the removeMapEventListener(EventListener) method.
void	**setCopyKey**(boolean copy) Sets whether or not the key needs to be copied when a map entry is created.
void	**setCopyMode**(CopyMode mode, Class valueInterface) Sets the CopyMode.
void	**setEvictionTriggers**(String evictionTriggers) Sets the types of additional eviction triggers, all evictors for the backing map will use the provided set of triggers.
void	**setEvictor**(Evictor e) Associates an Evictor with this BackingMap.
void	**setKeyOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data output format for all data access APIs that return cache keys.
void	**setLoader**(com.ibm.websphere.objectgrid.plugins.Loader loader) Associates a Loader with this BackingMap.
void	**setLockStrategy**(LockStrategy lockStrategy) Sets the LockStrategy.
void	**setLockTimeout**(int seconds) Sets the lock timeout used by the lock manager for this BackingMap.
void	**setMapEventListeners**(List eventListenerList) **Deprecated.** *This method is deprecated in version 7.1.1. Use the addMapEventListener(EventListener) or removeMapEventListener(EventListener) methods. Plugins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*
void	**setMapIndexPlugins**(List indexList) Sets the list of MapIndexPlugin objects for this BackingMap.
v	

void	**setNullValuesSupported**(boolean nullValuesSupported) Sets whether this BackingMap supports null values.
void	**setNumberOfBuckets**(int numBuckets) **Deprecated.** *Deprecated in 8.6. Buckets are no longer required. Use the isNearCacheEnabled() flag to disable the near cache in the ObjectGrid configuration XML file.*
void	**setNumberOfLockBuckets**(int numBuckets) Sets the number of lock buckets used by the lock manager for this BackingMap.
void	**setObjectTransformer**(com.ibm.websphere.objectgrid.plugins.ObjectTransformer t) Sets the ObjectTransformer object for use by this BackingMap and/or Loader.
void	**setOptimisticCallback**(com.ibm.websphere.objectgrid.plugins.OptimisticCallback checker) Sets the OptimisticCallback.
void	**setPreloadMode**(boolean async) Sets the preload mode if a Loader is set for this BackingMap.
void	**setReadOnly**(boolean readOnlyEnabled) Sets the map type of this BackingMap.
void	**setTimeBasedDBUpdateConfig**(com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig dbUpdateConfig) Set the time-based database update configuration object.
void	**setTimeToLive**(int seconds) Sets "time to live" of each map entry in seconds.
void	**setTtlEvictorType**(TTLType type) Sets how expiration time of a BackingMap entry is computed.
void	**setValueOutputFormat**(com.ibm.websphere.objectgrid.OutputFormat outputFormat) Sets the data output format for all data access APIs that return cache values.
void	**setWriteBehind**(String writeBehindParam) Enable write-behind updates for this map.

Methods inherited from interface com.ibm.websphere.objectgrid.ClientReplicableMap
disableClientReplication, enableClientReplication, getReplicationMode

Field Detail

DEFAULT_LOCK_TIMEOUT

`static final int` **`DEFAULT_LOCK_TIMEOUT`**

Default lock timeout used if `setLockTimeout(int)` is not invoked.

See Also:
Constant Field Values

DEFAULT_NUMBER_OF_BUCKETS

`static final int` **`DEFAULT_NUMBER_OF_BUCKETS`**

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*
Default number of lock buckets used if `setNumberOfBuckets(int)` is not invoked.

See Also:
Constant Field Values

DEFAULT_NUMBER_OF_LOCK_BUCKETS

`static final int` **`DEFAULT_NUMBER_OF_LOCK_BUCKETS`**

Default number of lock buckets used if `setNumberOfLockBuckets(int)` is not invoked.

See Also:
Constant Field Values

EVICTIONTRIGGER_MEMORY_USAGE_THRESHOLD

`static final String` **`EVICTIONTRIGGER_MEMORY_USAGE_THRESHOLD`**

The eviction trigger string constant to enable memory based eviction using memory usage threshold provided by the `java.lang.management.MemoryPoolMXBean`.

Since:
WAS XD 6.1.0.3
See Also:
`setEvictionTriggers(String)`, Constant Field Values

LOCAL

`static final int` **`LOCAL`**

Constant used to indicate this map is not a distributed map.

Since:
WAS XD 6.1
See Also:
`getMapType()`, Constant Field Values

SERVER

`static final int` **`SERVER`**

Constant used to indicate this map is a server map.

Since:
WAS XD 6.1
See Also:
`getMapType()`, Constant Field Values

CLIENT

`static final int` **`CLIENT`**

Constant used to indicate this map is a client to a server map

Since:
WAS XD 6.1
See Also:
`getMapType()`, Constant Field Values

Method Detail

getName

`String` **`getName`**`()`

Gets the name of the `BackingMap`.

Returns:
value specified when `BackingMap` was created.

---

setEvictor

`void` **`setEvictor`**`(Evictor e)`

Associates an `Evictor` with this `BackingMap`.

An `Evictor` aids with cleaning up the cache based on whatever algorithm is desired (LRU, LFU, etc). Passing `null` to this method removes a previously set `Evictor` object from an earlier invocation of this method.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

An Evictor that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous evictor which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

An `Evictor` may also implement the `BackingMapPlugin` interface in order to receive enhanced BackingMap plug-in lifecycle method calls. The plug-in is then also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:
`e` - Evictor instance
Throws:
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:
Evictor, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getEvictor

Evictor **getEvictor**()

Gets the Evictor being used by this BackingMap.

Returns:
the argument that was passed to the setEvictor(Evictor) method of this interface or null if setEvictor was not previously called for this BackingMap object.
See Also:
Evictor, setEvictor(Evictor)

---

setObjectTransformer

void **setObjectTransformer**(com.ibm.websphere.objectgrid.plugins.ObjectTransformer t)

Sets the ObjectTransformer object for use by this BackingMap and/or Loader.

An ObjectTransformer aids with the "serialization" of non-Serializable objects. It allows a custom copy function to be installed for more efficient object copy operations.

Note, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

An ObjectTransformer that implements the BackingMapLifecycleListener is automatically added as if the addMapEventListener(EventListener) method was called. Any previous transformer which implements BackingMapLifecycleListener is removed as if the removeMapEventListener(EventListener) method was called.

Parameters:
t - ObjectTransformer instance
Throws:
IllegalArgumentException - if the passed in ObjectTransformer is null
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.
See Also:
ObjectTransformer, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getObjectTransformer

com.ibm.websphere.objectgrid.plugins.ObjectTransformer **getObjectTransformer**()

Gets the ObjectTransformer object being used by this BackingMap and/or Loader.

Returns:
the argument that was passed to the setObjectTransformer(ObjectTransformer) method of this interface or the default ObjectTransformer object if the setObjectTransformer method was not previously called for this object.
See Also:
ObjectTransformer, setObjectTransformer(ObjectTransformer)

---

setOptimisticCallback

`void` **`setOptimisticCallback`**`(com.ibm.websphere.objectgrid.plugins.OptimisticCallback checker)`

Sets the `OptimisticCallback`.

The OptimisticCallback will be used to check the versions of cache entries during the commit phase. If no OptimisticCallback was previously set, a default OptimisticCallback will be used. For Entities, the default OptimisticCallback will use a version field that was specified in the entity metadata. For POJO objects or Entities that do not have a version field specified, the default OptimisticCallback uses the entire object as the version value. In order for it to work for POJO objects, the application's value object needs to have a useful equals(Object) method. If your application does not require versioning, but is using Optimistic locking, the NoVersioningOptimistCallback should be used.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

An OptimisticCallback that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous optimistic callback which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

Parameters:

`checker` - OptimisticCallback instance

Throws:

`IllegalArgumentException` - if the passed in `OptimisticCallback` is `null`

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`OptimisticCallback`, `NoVersioningOptimisticCallback`, `LockStrategy.OPTIMISTIC`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getOptimisticCallback

`com.ibm.websphere.objectgrid.plugins.OptimisticCallback` **`getOptimisticCallback`**`()`

Gets the `OptimisticCallback` being used by this `BackingMap` and/or `Loader` or `null` if the LockStrategy is not optimistic.

If no OptimisticCallback was previously set, a default OptimisticCallback will be used. For Entities, the default OptimisticCallback will use a version field that was specified in the entity metadata. For POJO objects or Entities that do not have a version field specified, the default OptimisticCallback uses the entire object as the version value. In order for it to work for POJO objects, the application's value object needs to have a useful equals(Object) method. If your application does not require versioning, but is using Optimistic locking, the NoVersioningOptimistCallback should be used.

Returns:

the argument that was passed to the `setOptimisticCallback(OptimisticCallback)` method of this interface or the default `OptimisticCallback` object if the `setOptimisticCallback` method was not previously called for this object. If Optimistic locking is not being used, this method will return null after ObjectGrid.initialize() has been invoked.

See Also:

`NoVersioningOptimisticCallback`, `OptimisticCallback`, `LockStrategy.OPTIMISTIC`, `setOptimisticCallback(OptimisticCallback)`

---

setLoader

`void` **`setLoader`**`(com.ibm.websphere.objectgrid.plugins.Loader loader)`

Associates a `Loader` with this `BackingMap`.

Only one Loader can be associated with a given `BackingMap`. Passing `null` to this method removes a previously set `Loader` object from an earlier invocation of this method and indicates that this `BackingMap` is not associated with a `Loader`.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

A loader that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous loader which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

A `Loader` may also implement the `BackingMapPlugin` interface in order to receive enhanced BackingMap plug-in lifecycle method calls. The plug-in is then also required to correctly implement each of the bean methods related to introspection of its state (for example `isInitialized()`, `isDestroyed()`, etc).

Parameters:

`loader` - Loader instance

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`Loader`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getLoader

`com.ibm.websphere.objectgrid.plugins.Loader` **`getLoader`**`()`

Gets the Loader being used by this `BackingMap`.

Returns:

the argument that was passed to the `setLoader(Loader)` method of this interface or `null` if `setLoader` was not previously called for this object.

See Also:

`Loader`, `setLoader(Loader)`

---

setPreloadMode

`void` **`setPreloadMode`**`(boolean async)`

Sets the preload mode if a `Loader` is set for this `BackingMap`.

If the parameter is true then the `Loader.preloadMap(Session, BackingMap)` is invoked asynchronously; otherwise it blocks the execution when loading data so the cache is unavailable until preload completes. Preloading occurs during `ObjectGrid` initialization.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`async` - If this is true then the cache is loaded asynchronously otherwise it blocks and the cache is unavailable until preload completes.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()`

method is called.

See Also:

`Loader.preloadMap(Session, BackingMap)`

---

getPreLoadMode

`boolean` **`getPreLoadMode`**`()`

Returns whether this `BackingMap` will be asynchronously preloaded or not if a `Loader` is set.

If true is returned then the `Loader.preloadMap(Session, BackingMap)` method is invoked asynchronously; otherwise it blocks the execution when loading data so the cache is unavailable until preload completes. Preloading occurs during `ObjectGrid` initialization.

Returns:

the argument that was passed to the `setPreloadMode(boolean)` method of this interface or `false` if setPreloadeMode was not previously called for this object.

See Also:

`Loader.preloadMap(Session, BackingMap)`, `setPreloadMode(boolean)`

---

addMapIndexPlugin

```
void addMapIndexPlugin(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index)
                       throws com.ibm.websphere.objectgrid.IndexAlreadyDefinedException
```

Adds an `MapIndexPlugin` to this Map. This method assumes the index implementation was constructed with the name of the attribute to index. The name of the index is specified when the index is constructed.

Note, to avoid an `IllegalStateException`, this method must be called prior to `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

A MapIndexPlugin that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous index which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

Parameters:

`index` - The index implementation.

Throws:

`IndexAlreadyDefinedException` - if this index already exists.

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`MapIndexPlugin`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getMapIndexPlugins

`List` **`getMapIndexPlugins`**`()`

Returns the current list of `MapIndexPlugin` objects for this BackingMap.

Returns:

The current list of `MapIndexPlugin`s for this BackingMap. The list is empty if the `addMapIndexPlugin(MapIndexPlugin)` or `setMapIndexPlugins(List)` method was not previously called for this BackingMap.

See Also:

addMapIndexPlugin(MapIndexPlugin), setMapIndexPlugins(List)

---

setMapIndexPlugins

void **setMapIndexPlugins**(List indexList)

Sets the list of MapIndexPlugin objects for this BackingMap. If the BackingMap already has a List of MapIndexPlugin objects, that list is replaced by the List passed as an argument to the current invocation of this method.

Note, to avoid an IllegalStateException, this method must be called prior to ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:

indexList - A non-null reference to a List of MapIndexPlugin objects.

Throws:

IllegalArgumentException - is thrown if indexList is null or the indexList contains either a null reference or an object that is not an instance of MapIndexPlugin.

See Also:

MapIndexPlugin, ObjectGrid.initialize(), ObjectGrid.getSession()

---

setCopyMode

void **setCopyMode**(CopyMode mode,
Class valueInterface)

Sets the CopyMode.

The CopyMode determines whether a get operation of an entry in the BackingMap returns the actual value, a copy of the value, or a proxy for the value. In the case of a proxy, the copy of the value does not occur unless a set method of the application provided value interface is invoked. It also determines that when a transaction is committed, whether a copy of the value object of an entry that was marked as dirty by the transaction is put into the BackingMap at commit time. The CopyMode does not specify if the object is copied when being read or written to a Loader. It is the responsibility of the implementor of a Loader to make copies as appropriate. The default CopyMode is CopyMode.COPY_ON_READ_AND_COMMIT.

Note, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:

mode - must be one of the final static variables defined in CopyMode. See CopyMode class for an explanation of each mode and how the valueInterface is used for CopyMode.COPY_ON_WRITE .

valueInterface - the value interface Class object. Specify null in version 7.1 and later.

Throws:

IllegalArgumentException - if mode is CopyMode.COPY_ON_WRITE and valueInterface parameter is null and CGLIB isn't in the classpath.

IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

See Also:

CopyMode, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getCopyMode

CopyMode **getCopyMode**()

Gets the `CopyMode` being used by this `BackingMap`.

Returns:
the argument that was passed to the `setCopyMode(CopyMode, Class)` method of this interface or the default `CopyMode` object if `setCopyMode` was not previously called for this object.

See Also:
CopyMode, setCopyMode(CopyMode, Class)

---

setLockStrategy

void **setLockStrategy**(LockStrategy lockStrategy)

Sets the `LockStrategy`.

The locking strategy represented by the `LockStrategy` object determines if the internal ObjectGrid lock manager is used whenever a map entry is accessed by a transaction. The default strategy is `LockStrategy.OPTIMISTIC`.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:
`lockStrategy` - must be one of the final static variables defined in `LockStrategy`. See `LockStrategy` class for an explanation of each locking strategy.

Throws:
IllegalStateException - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:
LockStrategy, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getLockStrategy

LockStrategy **getLockStrategy**()

Gets the `LockStrategy` object being used by this `BackingMap`.

Returns:
the argument that was passed to the `setLockStrategy(LockStrategy)` method of this interface or the default `LockStrategy` object if `setLockStrategy` was not previously called for this object.

See Also:
LockStrategy, setLockStrategy(LockStrategy)

---

setMapEventListeners

@Deprecated
void **setMapEventListeners**(List eventListenerList)

Deprecated. *This method is deprecated in version 7.1.1. Use the addMapEventListener(EventListener) or removeMapEventListener(EventListener) methods. Plugins that implement the ObjectGridLifecycleListener interface are automatically registered with the grid. Using this method will remove those automatically added listeners.*

Sets the list of `EventListener` objects.

If this BackingMap already has a List of EventListenerS, that list is replaced by the List passed as an argument to the current invocation of this method. This method can be called before and after the ObjectGrid.initialize() method.

Parameters:

eventListenerList - A non-null reference to a List of EventListener objects that are instances of BackingMapLifecycleListener or MapEventListener

Throws:

IllegalArgumentException - is thrown if eventListenerList is null, the eventListenerList contains either a null reference or an object that is not an instance of BackingMapLifecycleListener or MapEventListener

See Also:

EventListener, MapEventListener, BackingMapLifecycleListener, addMapEventListener(EventListener), removeMapEventListener(EventListener)

---

getMapEventListeners

List **getMapEventListeners**()

Gets the current list of EventListenerS.

Returns:

the current list of EventListener objects for this BackingMap.

See Also:

EventListener, MapEventListener, BackingMapLifecycleListener

---

addMapEventListener

void **addMapEventListener**(com.ibm.websphere.objectgrid.plugins.EventListener eventListener)

Adds an EventListener to this BackingMap.

Note, this method is allowed to be invoked before and after the ObjectGrid.initialize() method. Backing map plug-ins (Loader, Evictor, MapIndexPlugin, ObjectTransformer, OptimisticCallback) that implement the ObjectGridLifecycleListener are automatically added as listeners when added to the BackingMap.

Parameters:

eventListener - A non-null reference to a EventListener to add to the list. The listener must be an instance of BackingMapLifecycleListener or MapEventListener

Throws:

IllegalArgumentException - if eventListener is null or not an instance of BackingMapLifecycleListener or MapEventListener

See Also:

EventListener, MapEventListener, BackingMapLifecycleListener

---

addMapEventListener

void **addMapEventListener**(MapEventListener eventListener)

Deprecated. *This method is deprecated in version 7.1.1, use the addMapEventListener(EventListener) method.*

Provided for compatibility with old releases, use the addMapEventListener(EventListener) method.

Parameters:

eventListener - A non-null reference to a EventListener to add to the list. The listener must be an instance of BackingMapLifecycleListener or MapEventListener

removeMapEventListener

`void` **`removeMapEventListener`**`(com.ibm.websphere.objectgrid.plugins.EventListener eventListener)`

Removes an `EventListener` from this `BackingMap`.

Note, this method is allowed to be invoked before and after the `ObjectGrid.initialize()` method. Backing map plug-ins (`Loader`, `Evictor`, `MapIndexPlugin`, `ObjectTransformer`, `OptimisticCallback`) that implement the `ObjectGridLifecycleListener` are automatically removed as listeners when removed from the `ObjectGrid`.

Parameters:

`eventListener` - A non-null reference to an event listener that was previously added by invoking either the `addMapEventListener(EventListener)` or `setMapEventListeners(List)` method of this interface.

Throws:

`IllegalArgumentException` - if eventListener is null or not an instance of `BackingMapLifecycleListener` or `MapEventListener`

See Also:

`EventListener`, `MapEventListener`, `BackingMapLifecycleListener`, `addMapEventListener(EventListener)`

---

removeMapEventListener

`void` **`removeMapEventListener`**`(MapEventListener eventListener)`

Provided for compatibility with old releases, use the `removeMapEventListener(EventListener)` method.

Parameters:

`eventListener` - A non-null reference to an event listener that was previously added by invoking either the `addMapEventListener(EventListener)` or `setMapEventListeners(List)` method of this interface.

---

getPartitionId

`int` **`getPartitionId`**`()`

Gets the partition identifier being used by this `BackingMap`.

Returns:

The 0-based index for the partition represented by this `BackingMap` instance. If there is only a single partition defined for this `BackingMap` object, a 0 will be returned (default).

Since:

WAS XD 6.0.1

---

setReadOnly

`void` **`setReadOnly`**`(boolean readOnlyEnabled)`

Sets the map type of this `BackingMap`.

A map can be a read only map or a read/write map. Passing true as the parameter value will make this map a read only map; passing false as the parameter value will make this map a read/write map.

Note, to avoid an `IllegalStateException`, this method must be called prior to the

`ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`readOnlyEnabled` - If set to true, this `BackingMap` will be a read only map. If false, the map will be a read/write map.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

---

getReadOnly

`boolean` **`getReadOnly`**`()`

Retrieves the map type.

Returns:

the argument that was passed to `setReadOnly(boolean)` method of this interface. True is returned if this a read only map. A return value of false implies that this is a read/write map. If `setReadOnly` was never called, the default return value is false.

See Also:

`setReadOnly(boolean)`

---

getObjectGrid

`ObjectGrid` **`getObjectGrid`**`()`

Gets the `ObjectGrid` that owns this `BackingMap`.

Returns:

the ObjectGrid instance that owns this BackingMap.

See Also:

`ObjectGrid`

---

setNumberOfBuckets

`void` **`setNumberOfBuckets`**`(int numBuckets)`

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the `isNearCacheEnabled()` flag to disable the near cache in the ObjectGrid configuration XML file.*

Sets the number of buckets used by this `BackingMap`.

The `BackingMap` implementation uses a hash map for its implementation. If there are a lot of entries in the `BackingMap` then more buckets means better performance because the risk of collisions is lower as the number of buckets grows. More buckets also means more concurrency. If number of buckets is 0, no entries will be stored in the map, but the appropriate ObjectGrid and BackingMap plug-ins will still be called.

Once the `ObjectGrid` is initialized this parameter cannot be changed. Therefore, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`numBuckets` - The number of buckets to use.

Throws:

IllegalArgumentException - if numBuckets is less than 0.
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

See Also:

ObjectGrid.initialize(), ObjectGrid.getSession()

---

getNumberOfBuckets

int **getNumberOfBuckets**()

Deprecated. *Deprecated in 8.6. Buckets are no longer required. Use the isNearCacheEnabled() flag to disable the near cache in the ObjectGrid configuration XML file.*

Gets the number of buckets defined for this BackingMap.

Returns:

the same value passed to the setNumberOfBuckets(int) method or DEFAULT_NUMBER_OF_BUCKETS if setNumberOfBuckets was never called.

See Also:

setNumberOfBuckets(int), DEFAULT_NUMBER_OF_BUCKETS

---

setNumberOfLockBuckets

void **setNumberOfLockBuckets**(int numBuckets)

Sets the number of lock buckets used by the lock manager for this BackingMap.

When LockStrategy.OPTIMISTIC or LockStrategy.PESSIMISTIC is used for this BackingMap, a lock manager is created for the BackingMap. The lock manager uses a hash map to keep track of entries that are locked by 1 or more transactions. If there are a lot of entries in the hash map, then more lock buckets means better performance as the risk of collisions is lower as the number of buckets grows. More lock buckets also means more concurrency. When the lock strategy is LockStrategy.NONE, no lock manager is used by this BackingMap. In this case, a call to this method does nothing.

Once the ObjectGrid is initialized, the number of lock buckets cannot be changed. Therefore, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:

numBuckets - The number of lock buckets to use.

Throws:

IllegalArgumentException - if numBuckets is less than 1.
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

See Also:

LockStrategy, ObjectGrid.initialize(), ObjectGrid.getSession()

---

getNumberOfLockBuckets

int **getNumberOfLockBuckets**()

Gets the number of lock buckets defined for the hash map used by lock manager for this backing map.

Returns:

the same value passed to the `setNumberOfLockBuckets(int)` method or `DEFAULT_NUMBER_OF_LOCK_BUCKETS` if `setNumberOfLockBuckets` was never called.

See Also:

`setNumberOfLockBuckets(int)`, `DEFAULT_NUMBER_OF_LOCK_BUCKETS`

---

setLockTimeout

`void` **`setLockTimeout`**`(int seconds)`

Sets the lock timeout used by the lock manager for this `BackingMap`.

When `LockStrategy.OPTIMISTIC` or `LockStrategy.PESSIMISTIC` is used for this `BackingMap`, a lock manager is created for the `BackingMap`. To prevent deadlocks from occurring, the lock manager has a default timeout value for waiting for a lock to be granted. If this timeout limit is exceeded, a `LockTimeoutException` is thrown. The default value of `DEFAULT_LOCK_TIMEOUT` should be sufficient for most applications, but on a heavily loaded system, a timeout may occur when no deadlock exists. In that case, this method can be used to increase the lock timeout value from the default to whatever is desired to prevent false timeout exceptions from occurring. When the lock strategy is `LockStrategy.NONE`, no lock manager is used by this `BackingMap`. In this case, a call to this method does nothing. A lock timeout value of zero indicates to not wait for the lock if it is not immediately available.

Once the lock manager is initialized, the lock timeout value cannot be changed. Therefore, to avoid an `IllegalStateException`, this method must be called prior to `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application. When an entry is fetched the lock timeout can be changed for a given transaction using `ObjectMap.setLockTimeout(int)`

Parameters:

`seconds` - is the lock timeout value to use in seconds.

Throws:

`IllegalArgumentException` - if seconds is less than 0.

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`DEFAULT_LOCK_TIMEOUT`, `LockStrategy`, `LockTimeoutException`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`, `ObjectMap.setLockTimeout(int)`

---

getLockTimeout

`int` **`getLockTimeout`**`()`

Gets the lock timeout value used by the lock manager for this `BackingMap`.

Returns:

the same value passed to the `setLockTimeout(int)` method or `DEFAULT_LOCK_TIMEOUT` if `setLockTimeout` was never called.

See Also:

`DEFAULT_LOCK_TIMEOUT`, `setLockTimeout(int)`

---

setNullValuesSupported

`void` **`setNullValuesSupported`**`(boolean nullValuesSupported)`

Sets whether this `BackingMap` supports `null` values.

If `null` values are supported, users need to be careful when a `get` operation returns a `null`

reference. It could be due to the fact that the key is not found in the `BackingMap`, or that the value in the `BackingMap` is `null`. To determine if a key was not found, or the value is `null`, the `containsKey` method can be used.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`nullValuesSupported` - If set to true, `null` values are supported; otherwise `null` values are not supported.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`ObjectGrid.initialize()`, `ObjectGrid.getSession()`, `ObjectMap.containsKey(Object)`

---

getNullValuesSupported

`boolean **getNullValuesSupported**()`

Gets whether this `BackingMap` supports `null` values or not.

Returns:

the same value passed to the `setNullValuesSupported(boolean)` method or the default value of true if `setNullValuesSupported` was never called.

See Also:

`setNullValuesSupported(boolean)`

---

setCopyKey

`void **setCopyKey**(boolean copy)`

Sets whether or not the key needs to be copied when a map entry is created.

Copying the key object allows the application to use the same key object for each ObjectMap operation. The application changes the key object state prior to each `ObjectMap` operation so that it can work with different entries using the same key object. If a separate key object is used for each entry, then there is no reason to copy the key object. This attribute allows an application to make the tradeoff of copying key object versus using more memory as a result of separate key object used by the application for each entry. If this method is not called, then the default of `false` is used (e.g. the key is NOT copied).

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`copy` - If `true` is specified, then this `BackingMap` uses the `ObjectTransformer.copyKey(Object)` method to copy the key object when necessary.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`ObjectGrid.initialize()`, `ObjectGrid.getSession()`, `ObjectTransformer.copyKey(Object)`

---

getCopyKey

`boolean` **`getCopyKey`**`()`

Gets whether keys are copied for this `BackingMap`.

Returns:
the same value passed to the `setCopyKey(boolean)` method or the default value of false if `setCopyKey` was never called.

See Also:
`setCopyKey(boolean)`

---

setTimeToLive

`void` **`setTimeToLive`**`(int seconds)`

Sets "time to live" of each map entry in seconds.

If this method is not called, the lifetime of an entry is forever (or until the application explicitly removes or invalidates the entry, or a user defined `Evictor` evicts the entry). Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:
`seconds` - the number of seconds a map entry is allowed to live in map before being evicted.

Throws:
`IllegalArgumentException` - if seconds is less than 0.
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:
`setTtlEvictorType(TTLType)`, `ObjectMap.setTimeToLive(int)`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getTimeToLive

`int` **`getTimeToLive`**`()`

Gets the number of seconds for an entry to live.

This value returned is in seconds and 0 indicates forever.

Returns:
the same value passed to the `setTimeToLive(int)` method or 0 if `setLockTimeout` was never called.

See Also:
`setTimeToLive(int)`

---

setTtlEvictorType

`void` **`setTtlEvictorType`**`(TTLType type)`

Sets how expiration time of a `BackingMap` entry is computed.

If this method is not called, `TTLType.NONE` is used to indicate the map entry has no expiration time (e.g. is allowed to live until explicitly removed or invalidated by the application, or evicted by a user defined `Evictor`).

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`type` - must be one of the public constants declared in the `TTLType` class.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

See Also:

`TTLType`, `ObjectGrid.initialize()`, `ObjectGrid.getSession()`

---

getTtlEvictorType

`TTLType` **`getTtlEvictorType`**`()`

Gets how expiration time of a `BackingMap` entry is computed.

Returns:

the `TTLType` that was passed to the `setTtlEvictorType(TTLType)` or `TTLType.NONE` if `setTtlEvictorType` was never called.

See Also:

`setTtlEvictorType(TTLType)`, `TTLType`

---

createDynamicIndex

```
void createDynamicIndex(String name,
                        boolean isRangeIndex,
                        String attributeName,
                        com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicInd
exCallback)
                        throws com.ibm.websphere.objectgrid.IndexAlreadyDefinedException,
                            IllegalArgumentException
```

Creates a dynamic index on the `BackingMap`.

Required Client Permission: `ServerMapPermission.REPLICATE`

Parameters:

`name` - the name of the index. The name can not be null or a zero length string.
`isRangeIndex` - Indicate whether to create a `MapRangeIndex` or a `MapIndex`. If set to true, the index will be a type of `MapRangeIndex`.
`attributeName` - The name of the attribute to be indexed. The attributeName can not be null or a zero length string.
`dynamicIndexCallback` - The callback that will invoke upon dynamic index events. The dynamicIndexCallback is optional and can be null.

Throws:

`IllegalArgumentException` - if name or attributeName is null or a zero length string.
`IndexAlreadyDefinedException` - if a `MapIndexPlugin` with the specified name already exists.

Since:

WAS XD 6.0.1

See Also:

`MapIndex`, `MapIndexPlugin`, `MapRangeIndex`, `ObjectMap.getIndex(String)`

---

createDynamicIndex

```
void createDynamicIndex(com.ibm.websphere.objectgrid.plugins.index.MapIndexPlugin index,
```

```
                    com.ibm.websphere.objectgrid.plugins.index.DynamicIndexCallback dynamicInd
exCallback)
                    throws com.ibm.websphere.objectgrid.IndexAlreadyDefinedException,
                           IllegalArgumentException
```

Creates a dynamic index on the `BackingMap`.

Required Client Permission: `ServerMapPermission.DYNAMIC_INDEX`

A MapIndexPlugin that implements the `BackingMapLifecycleListener` is automatically added as if the `addMapEventListener(EventListener)` method was called. Any previous index which implements `BackingMapLifecycleListener` is removed as if the `removeMapEventListener(EventListener)` method was called.

Parameters:

`index` - The index implementation. The index can not be `null`.
`dynamicIndexCallback` - The callback that will invoke upon dynamic index events. The dynamicIndexCallback is optional and can be null.

Throws:

`IllegalArgumentException` - if index is `null` or `index.getName()` returns `null` or a zero length string.
`IndexAlreadyDefinedException` - if a `MapIndexPlugin` with the specified name already exists.

Since:

WAS XD 6.0.1

See Also:

`MapIndexPlugin`, `ObjectMap.getIndex(String)`

---

removeDynamicIndex

```
void removeDynamicIndex(String name)
                    throws com.ibm.websphere.objectgrid.IndexUndefinedException,
                           IllegalArgumentException
```

Removes a dynamic index on the `BackingMap`.

Required Client Permission: `ServerMapPermission.DYNAMIC_INDEX`

Parameters:

`name` - the name of the index. The name can not be `null`.

Throws:

`IllegalArgumentException` - if name is `null`.
`IndexUndefinedException` - if a `MapIndexPlugin` with the specified name does not exists.

Since:

WAS XD 6.0.1

See Also:

`createDynamicIndex(MapIndexPlugin, DynamicIndexCallback)`, `createDynamicIndex(String, boolean, String, DynamicIndexCallback)`

---

getPartitionManager

```
PartitionManager getPartitionManager()
```

Allows access to the `PartitionManager` that is defined for this `BackingMap`. This access may be useful for `Loaders during Loader.preloadMap(Session, BackingMap) processing (to properly partition the data to be loaded).`

Returns:

PartitionManager associated with this BackingMap.

Since:

WAS XD 6.0.1

See Also:

PartitionManager, Loader.preloadMap(Session, BackingMap)

---

getEntityMetadata

com.ibm.websphere.projector.md.EntityMetadata **getEntityMetadata**()

Retreive the metadata for the entity associated with this backing map.

Returns:
the EntityMetadata if an entity is associated with this backing map or null if there is no entity associated with this backing map.

Since:
WAS XD 6.1

---

setWriteBehind

void **setWriteBehind**(String writeBehindParam)

Enable write-behind updates for this map.

If a map is configured with write-behind loader update, the updates (could be insert type, remove type, or update type) to the backend are not instantly updated to the back end by calling the Loader.batchUpdate(TxID, LogSequence) method. Instead, they are queued in a write-behind queue map and updated to the back end periodically.

A write-behind update is pushed to the backend periodically within a different transaction from the one the update is made to ObjectGrid. When the write-behind update to the backend fails, for example, due to data integrity problem, it is too late to roll back the original ObjectGrid transaction. ObjectGrid will invalidate the entry and create an entry in a failed database update map. The name of this failed database update map is WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX+baseMapName. The key of the entry in this map is an auto-increment Integer, and the value is a LogElement. The logElement can be used to compensate the failure.

Depending on your grid use case and your back end loader configuration, your back end loader or the back end database might benefit from having upsert operations instead of insert and update operations in the LogElements that it receives for a transaction. Use the ConvertToUpsert=true configuration option on the writeBehindParam to have the write behind loader convert insert and update LogElement operations to upsert LogElement operations when they are passed to the back end loader. Not all back end loaders may support the upsert operation, be certain that the back end loader supports upsert operations before using the ConvertToUpsert=true clause in the writeBehindParam. The default value is ConvertToUpsert=false.

Note, to avoid an IllegalStateException, this method must be called prior to the ObjectGrid.initialize() method. Also, keep in mind that the ObjectGrid.getSession() method implicitly calls the ObjectGrid.initialize() method if it has yet to be called by the application.

Parameters:
writeBehindParam - a write-behind parameter consisting of a maximum update time and/or a maximum key update count. The format of the write-behind parameter is "T[time];C[count][;ConvertToUpsert=true]", for example, "T100;C2000". "T100;C2000" means the loader will write to the back end when there are 2000 pending keys to be updated or when 100 seconds have passed since the last update. The default update time is 300 seconds and the default update key count is 1000. You can configure the update time only, the update key count only, or an empty string. The default value(s) will then be used in either of the above three cases. The default value is null to disable write-behind updates.

Throws:
IllegalArgumentException - if the write behind parameters are unknown or improperly formatted.
IllegalStateException - if this method is called after the ObjectGrid.initialize() method is called.

Since:

WAS XD 6.1.0.3

See Also:

`WriteBehindLoaderConstants`

---

getWriteBehind

`String` **`getWriteBehind`**`()`

Get the write-behind parameter. A write-behind parameter consists of a maximum update time and/or a maximum key update count. The format of the write-behind parameter is "T[time];C[count][;ConvertToUpsert=true]".

Returns:

the write-behind parameter. If the write-behind parameter is not set, `null` will be returned.

Since:

WAS XD 6.1.0.3

See Also:

`setWriteBehind(String)`

---

setTimeBasedDBUpdateConfig

`void` **`setTimeBasedDBUpdateConfig`**`(com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig dbUpdateConfig)`

Set the time-based database update configuration object.

When a time-based database update configuration object is set, a thread will be started automatically to update or invalidate the ObjectGrid maps with the latest updates (inserts and updates) from the database.

For a local ObjectGrid map, the thread will be launched in the same JVM. For a distributed ObjectGrid map in an ObjectGrid container, the thread will be automatically launched in partition 0. No database update thread will be started in a client side near cache.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:

`dbUpdateConfig` - the time-based database update configuration object or `null`.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

Since:

WAS XD 6.1.0.3

---

getTimeBasedDBUpdateConfig

`com.ibm.websphere.objectgrid.TimeBasedDBUpdateConfig` **`getTimeBasedDBUpdateConfig`**`()`

Get the time-based database update configuration object.

Returns:

the time-based database update configuration object or `null` if not set.

Since:

WAS XD 6.1.0.3

See Also:
setTimeBasedDBUpdateConfig(TimeBasedDBUpdateConfig)

---

getMapType

`int` **`getMapType`**`()`

Returns the type of BackingMap.

The return value is equivalent to one of the constants declared on this interface, `LOCAL`, `SERVER`, or `CLIENT`.

Returns:
the map type
Since:
WAS XD 6.1

---

getEvictionTriggers

`String` **`getEvictionTriggers`**`()`

Returns the types of additional eviction triggers.

The available eviction trigger strings are a described in the String constants in this interface that begin with the name: `EVICTIONTRIGGER`.

Returns:
a semicolon separated list of eviction triggers
Since:
WAS XD 6.1.0.3

---

setEvictionTriggers

`void` **`setEvictionTriggers`**`(String evictionTriggers)`

Sets the types of additional eviction triggers, all evictors for the backing map will use the provided set of triggers.

The available eviction trigger strings are a described in the String constants in this interface that begin with the name: `EVICTIONTRIGGER`.

Note, to avoid an `IllegalStateException`, this method must be called prior to the `ObjectGrid.initialize()` method. Also, keep in mind that the `ObjectGrid.getSession()` method implicitly calls the `ObjectGrid.initialize()` method if it has yet to be called by the application.

Parameters:
`evictionTriggers` - a semicolon separated list of eviction triggers
Throws:
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.
`IllegalArgumentException` - if the eviction triggers are unknown or improperly formatted.
Since:
WAS XD 6.1.0.3

---

getMapSetName

`String` **`getMapSetName`**()

Retrieves the name of the MapSet that this BackingMap is currently associated with. A "null" return value indicates it is currently not associated with a MapSet. This method will only return a non null value for a client or server map.

Returns:
name of associated MapSet
Since:
7.1
See Also:
`getMapType()`

---

getSerializerAccessor

`com.ibm.websphere.objectgrid.plugins.io.SerializerAccessor` **`getSerializerAccessor`**()

Retrieve the SerializerAccessor for this map.

Returns:
the SerializerAccessor
Since:
7.1.1

---

getState

`com.ibm.websphere.objectgrid.plugins.BackingMapLifecycleListener.State` **`getState`**()

Retrieve the current life cycle state of this map.

Returns:
the current state.
Since:
7.1.1

---

isNearCacheInvalidationEnabled

`boolean` **`isNearCacheInvalidationEnabled`**()

If true, clients with local caches are automatically invalidated when the data grid map is updated.

Returns:
true if client near cache invalidation is enabled.
Since:
8.6, XC10 2.5

---

isNearCacheLastAccessTTLSyncEnabled

`boolean` **`isNearCacheLastAccessTTLSyncEnabled`**()

If true, clients automatically send time-to-live access information to the remote data grid when accessed and the `TTLType.LAST_ACCESS_TIME` TTL evictor type is configured.

Returns:
true if last-access time-to-live information is sent to the remote data grid.
Since:
8.6, XC10 2.5

isNearCacheEnabled

`boolean` **`isNearCacheEnabled`**`()`

If true, the client near cache is enabled for supported configurations. The client near cache can only be enabled when using optimistic locking or when locking is disabled.

Returns:
true if the client near cache is enabled.

---

getKeyOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getKeyOutputFormat`**`()`

Retrieves the data format for all data access APIs that return cache keys.

This value does not reflect the data output format that plug-ins will see. See the `PluginOutputFormat` annotation for details on how to influence the data object format that plug-ins receive.

Returns:
the data output format.
Since:
8.6, XC10 2.5

---

setKeyOutputFormat

`void` **`setKeyOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`

Sets the data output format for all data access APIs that return cache keys.

When set to `OutputFormat.UNDEFINED`, the key output format defaults to `OutputFormat.RAW` when using a custom `KeyDataSerializer` plug-in. The key output format is `OutputFormat.NATIVE` in all other cases.

Parameters:
`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default.
Throws:
`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.
Since:
8.6, XC10 2.5

---

getValueOutputFormat

`com.ibm.websphere.objectgrid.OutputFormat` **`getValueOutputFormat`**`()`

Retrieves the data output format for all data access APIs that return cache values.

This value does not reflect the data output format that plug-ins will see. See the `PluginOutputFormat` annotation for details on how to influence the data object format that plug-ins receive.

Returns:
the data output format or `OutputFormat.UNDEFINED` if the default should be used.
Since:
8.6, XC10 2.5

setValueOutputFormat

`void` **`setValueOutputFormat`**`(com.ibm.websphere.objectgrid.OutputFormat outputFormat)`

Sets the data output format for all data access APIs that return cache values.

When set to `OutputFormat.UNDEFINED`, the value output format defaults to `OutputFormat.RAW` when using a custom `ValueDataSerializer` plug-in or when the `CopyMode.COPY_TO_BYTES_RAW` CopyMode set.

The value output format is `OutputFormat.NATIVE` in all other cases.

Parameters:

`outputFormat` - the data output format to use or `OutputFormat.UNDEFINED` to use the default.

Throws:

`IllegalStateException` - if this method is called after the `ObjectGrid.initialize()` method is called.

Since:

8.6, XC10 2.5

com.ibm.websphere.objectgrid

Class AvailabilityState

java.lang.Object
  com.ibm.websphere.objectgrid.AvailabilityState

All Implemented Interfaces:

Serializable

```
public final class AvailabilityState
extends Object
implements Serializable
```

Each shard in a distributed ObjectGrid has an availability state associated with it. This state refers to the shard's ability to process incoming requests.

Since:

WAS XD 6.1.0.3, XC10

See Also:

Serialized Form

Field Summary

static AvailabilityState	**OFFLINE** An AvailabilityState.OFFLINE indicates that a shard is offline and unable to process requests.
static AvailabilityState	**ONLINE**

[bilityState](#)	An AvailabilityState.ONLINE indicates that a shard is online.
static [AvailabilityState](#)	**PRELOAD** An AvailabilityState.PRELOAD indicates that a shard is in the preload state.
static [AvailabilityState](#)	**QUIESCE** An AvailabilityState.QUIESCE indicates that a shard is in quiesce.
static [AvailabilityState](#)	**UNKNOWN** An AvailabilityState.UNKNOWN indicates that the availability state of the shard could not be determined.

State	

Method Summary

int	**getId**() Returns the internal identifier for this state.
String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

OFFLINE

```
public static final AvailabilityState OFFLINE
```

An AvailabilityState.OFFLINE indicates that a shard is offline and unable to process requests.

PRELOAD

```
public static final AvailabilityState PRELOAD
```

An AvailabilityState.PRELOAD indicates that a shard is in the preload state. When in the preload state, a shard will reject all requests that are not initiated from a client that is preloading data into the ObjectGrid.

ONLINE

```
public static final AvailabilityState ONLINE
```

An AvailabilityState.ONLINE indicates that a shard is online. A shard that is online is available for processing requests.

QUIESCE

```
public static final AvailabilityState QUIESCE
```

An AvailabilityState.QUIESCE indicates that a shard is in quiesce. Quiesce is a transitional state. Shards that are in the quiesce state are on their way to being offline. A shard in the quiesce state will allow all pending transactions to complete before moving to the AvailabilityState.OFFLINE, assuming that all pending transactions complete within 30 seconds after entering the quiesce state.

UNKNOWN

`public static final AvailabilityState UNKNOWN`

An AvailabilityState.UNKNOWN indicates that the availability state of the shard could not be determined.

Method Detail

getId

`public int getId()`

Returns the internal identifier for this state.

Returns:
the internal id.

Since:
7.1.1

---

toString

`public String toString()`

Overrides:
`toString` in class `Object`

Package com.ibm.websphere.xs.ra

These are the eXtreme Scale Resource Adapter APIs that allows integration with a Java EE application.

See:

Description

Interface Summary

ObjectGridJ2CConnectionMBean	Allows administration clients control the lifecycle of the eXtreme Scale resource adapter's connection.

Class Summary

XSConnection	An XSConnection represents an application level connection handle to an eXtreme Scale ObjectGrid.
XSConnectionFactory	The XSConnectionFactory creates connections to eXtreme Scale ObjectGrids.
XSConnectionSpec	The ConnectionSpec for retrieving XSConnections from an XSConnectionFactory
XSManagedConnectionFactory	The XSManagedConnectionFactory create managed connections to the eXtreme Scale ObjectGrids.
XSResourceAdapter	The eXtreme Scale resource adapter

Package com.ibm.websphere.xs.ra Description

These are the eXtreme Scale Resource Adapter APIs that allows integration with a Java EE application.

Introduction

The eXtreme Scale Resource Adapter provides client connection management and local transaction support, allowing Java EE applications to look-up eXtreme Scale client connections and demarcate transactions using the `LocalTransaction` interface or the `Session` interface.

When used with WebSphere Application Server with last participant support enabled, the eXtreme Scale transaction can be enlisted in a global transaction as the last, single-phase participant.

Programming Tutorial

The following sections show snippets on the usage of the ConnectionFactory. The resource

adapter is is JCA 1.5 compliant. To use the resource adapter the following steps must be followed:

1. Install the resource adapter
2. Configure a J2C ConnectionFactory
3. Configure a javax.resource.cci.ConnectionFactory resource reference in the application.
4. Look-up the `XSConnectionFactory`
5. Choose one of the following transaction options:

   Use auto-commit, local transactions:

   1. Retrieve a `XSConnection`
   2. Retrieve and use the `Session` to interact with the data grid.
   3. Close the connection.

   Use an ObjectGrid Session to demarcate a local transaction:

   1. Retrieve a `XSConnection`
   2. Retrieve the `Session`
   3. Use the Session.begin() method to start the transaction.
   4. Use the Session to interact with the data grid.
   5. Use the Session.commit() or rollback()) methods to end the transaction.
   6. Close the connection.

   Use a javax.resource.cci.LocalTransction to demarcate a local transaction:

   1. Retrieve a `XSConnection`
   2. Retrieve the javax.resource.cci.LocalTransaction using the XSConnection.getLocalTransaction() method.
   3. Use the LocalTransaction.begin() method to start the transaction.
   4. Retrieve and use the `Session` to interact with the data grid.
   5. Use the LocalTransaction.commit() or rollback() methods to end the transaction.
   6. Close the connection.

   Enlist the connection in a global transaction:

   1. Lookup the UserTransction.
   2. Begin the global transaction
   3. Retrieve a `XSConnection`
   4. Retrieve and use the `Session`
   5. Close the connection.
   6. Commit or rollback the global transaction.

Installing the resource adapter

The eXtreme Scale resource adapter is included in the wxsra.rar resource adapter archive with the eXtreme Scale product. See the WebSphere eXtreme Scale version 8.5 information center (or later) for details on how to install and configure the resource adapter.

Configuring the J2C ConnectionFactory

The eXtreme Scale resource adapter allows configuring one or J2C ManagedConnectionFactory instances. Each ManagedConnectionFactory is managed by the application server and represents a connection to a single catalog service domain. the ManagedConnectionFactory can include the name of the data grid, or the data grid can be provided when the connection is retrieved by the application.

The ManagedConnectionFactory is configured in the application server configuration, bound to a global JNDI name, and provides the following configuration properties:

ConnectionNa	

me	(Optional) The name of the eXtreme Scale client connection.
CatalogServiceEndpoints	The catalog service domain end points in the form: <host>:<port>,<host><port>. Required if catalog service domain is not set.
CatalogServiceDomain	The catalog service domain name. Required if catalog service end points are not set.
ObjectGridName	(Optional) The data grid name. If not specified, the client must provide the data grid name when retrieving the connection resource.
ObjectGridURL	(Optional) The URL of the client data grid, override XML file.
ObjectGridResource	(Optional) The resource path of the client data grid, override XML file.
ClientPropertiesURL	(Optional) The URL of the client properties file.
ClientPropertiesResource	(Optional) The resource path of the client properties file.

Obtaining a ConnectionFactory instance.

The Java EE application can use resource injection to inject a ConnectionFactory resource into the application, or it can be looked-up using a resource reference. The Java EE application must first configure resource reference for a javax.resource.cci.ConnectionFactory.

For example:

```
InitialContext ctx = new InitialContext();
XSConnectionFactory cf = (XSConnectionFactory) ctx.lookup("java:comp/env/wxsconnection");
```

Retrieving a Connection

After the `XSConnectionFactory` has been looked-up or injected into the application, use one of the getConnection() methods to retrieve a client connection to the data grid. The connection will automatically be established when the first connection is retrieved and will be maintained until the resource adapter is stopped or the connection is reset using the ObjectGridJ2CConnection management bean.

For example:

```
XSConnection con = cf.getConnection("MyGrid");
```

Obtaining an ObjectGrid Session instance.

The XSConnection provides a getSession() method that gives the application direct access to the ObjectGrid Session. The Session is used to interact with the data grid and is valid for the life of the XSConnection. The XSConnection is a handle to a connection and becomes invalid after the application context completes per the Java EE specification.

An eXtreme Scale local transaction can be driven by the Session, javax.resource.cci.LocalTransaction or a global transaction. The transaction methods cannot be mixed.

Closing a connection

After the application has finished using the connection, the connection must be closed. The Java EE container typically will also close the connection automatically at the appropriate times. When the connection is closed the Session and any other objects retrieved directlly or indirectly from the connection become invalid.

---

com.ibm.websphere.xs.ra

Class XSManagedConnectionFactory

java.lang.Object
  com.ibm.websphere.xs.ra.XSManagedConnectionFactory

All Implemented Interfaces:

Serializable, ManagedConnectionFactory, ResourceAdapterAssociation

---

```
public final class XSManagedConnectionFactory
extends Object
implements ManagedConnectionFactory, ResourceAdapterAssociation
```

The XSManagedConnectionFactory create managed connections to the eXtreme Scale ObjectGrids.

Since:

8.5, XC10

See Also:

Serialized Form

Constructor Summary

XSManagedConnectionFactory()

Method Summary

Object	**createConnectionFactory**() Creates a non-managed XSConnectionFactory instance.
Object	**createConnectionFactory**(ConnectionManager conMgr) Creates a managed XSConnectionFactory instance.
ManagedConnection	**createManagedConnection**(Subject subject, ConnectionRequestInfo conReq) Creates a new physical connection to the underlying ObjectGrid

[ction](#)	
void	**destroy**() Destroy this MCF and release any other resources.
boolean	**equals**(Object obj)
String	**getCatalogServiceDomain**() Get the eXtreme Scale specific property: CatalogDomain
String	**getCatalogServiceEndpoints**() Get the eXtreme Scale specific property: CatalogServiceEndpoints
String	**getClientPropertiesResource**() Get the eXtreme Scale specific property: ClientPropertiesResource
String	**getClientPropertiesURL**() Get the eXtreme Scale specific property: ClientPropertiesURL
String	**getConnectionName**() Get the eXtreme Scale specific property: ConnectionName
PrintWriter	**getLogWriter**() Get the log writer for this XSManagedConnectionFactory instance.
String	**getObjectGridName**() Get the eXtreme Scale specific property: ObjectGridName

String	**getObjectGridResource**() Get the eXtreme Scale specific property: ObjectGridResource
String	**getObjectGridURL**() Get the eXtreme Scale specific property: ObjectGridURL
ResourceAdapter	**getResourceAdapter**() Retrieve the associated eXtreme Scale resource adapter instance.
int	**hashCode**()
boolean	**isLocalGrid**() Retrieve the eXtreme Scale specific property: LocalGrid
ManagedConnection	**matchManagedConnections**(Set connectionSet, Subject subject, ConnectionRequestInfo cxRequestInfo) Returns a matched connection from the candidate set of connections.
void	**setCatalogServiceDomain**(String catalogServiceDomain) Set the eXtreme Scale specific property: CatalogDomain
void	**setCatalogServiceEndpoints**(String catalogServiceEndpoints) Set the eXtreme Scale specific property: CatalogServiceEndpoints
vo	**setClientPropertiesResource**(String clientPropsResource) Get the eXtreme Scale specific property: ClientPropertiesResource

void	**setClientPropertiesURL**(String clientPropsURL) Set the eXtreme Scale specific property: ClientPropertiesURL
void	**setConnectionName**(String connName) Set the eXtreme Scale specific property: ConnectionName
void	**setLocalGrid**(boolean useLocal) Set the eXtreme Scale specific property: LocalGrid
void	**setLogWriter**(PrintWriter out) Set the log writer for this ManagedConnectionFactory instance
void	**setObjectGridName**(String objectGridName) Set the eXtreme Scale specific property: ObjectGrid Name
void	**setObjectGridResource**(String objectGridResource) Set the eXtreme Scale specific property: ObjectGridResource
void	**setObjectGridURL**(String objectGridURL) Set the eXtreme Scale specific property: ObjectGridURL
void	**setResourceAdapter**(ResourceAdapter ra) Associate this object with an eXtreme Scale resource adapter.
String	**toString**()

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

XSManagedConnectionFactory

public **XSManagedConnectionFactory**()

Method Detail

getResourceAdapter

public ResourceAdapter **getResourceAdapter**()

Retrieve the associated eXtreme Scale resource adapter instance.

Specified by:
`getResourceAdapter` in interface `ResourceAdapterAssociation`

Returns:
The associated eXtreme Scale resource adapter

See Also:
`ResourceAdapterAssociation.getResourceAdapter()`

---

setResourceAdapter

```
public void setResourceAdapter(ResourceAdapter ra)
                        throws ResourceException
```

Associate this object with an eXtreme Scale resource adapter. Note, this method must be called exactly once. That is, the association must not change during the lifetime of this object.

Specified by:
`setResourceAdapter` in interface `ResourceAdapterAssociation`

Parameters:
`ra` - the eXtreme Scale resource adapter

Throws:
`ResourceException`
`IllegalStateException`

See Also:
`ResourceAdapterAssociation.setResourceAdapter(ResourceAdapter)`

---

createConnectionFactory

```
public Object createConnectionFactory()
                                throws ResourceException
```

Creates a non-managed XSConnectionFactory instance. The XSConnectionFactory instance gets initialized with a default XSConnectionManager instance provided by the eXtreme Scale resource adapter.

Specified by:
`createConnectionFactory` in interface `ManagedConnectionFactory`

Returns:
the new XSConnectionFactory instance

Throws:
`ResourceException`

See Also:
`ManagedConnectionFactory.createConnectionFactory()`

---

createConnectionFactory

```
public Object createConnectionFactory(ConnectionManager conMgr)
                                throws ResourceException
```

Creates a managed XSConnectionFactory instance. The XSConnectionFactory instance gets initialized with the passed ConnectionManager. In the managed scenario, ConnectionManager is provided by the application server.

Specified by:
`createConnectionFactory` in interface `ManagedConnectionFactory`

Parameters:

`conMgr` - - Connection Manager to be associated with the XSConnectionFactory.

Returns:

the new XSConnectionFactory instance

Throws:

`ResourceException` - ResourceAdapterInternalException

See Also:

`ManagedConnectionFactory.createConnectionFactory(ConnectionManager)`

---

createManagedConnection

```
public ManagedConnection createManagedConnection(Subject subject,
                                                 ConnectionRequestInfo conReq)
                                          throws ResourceException
```

Creates a new physical connection to the underlying ObjectGrid

Specified by:

`createManagedConnection` in interface `ManagedConnectionFactory`

Parameters:

`subject` - - Caller's security information. WebSphere eXtreme Scale client uses the credentials specified in a CredentialGenerator as part of the ConnectionSpec or the client properties.

`conReq` - - XS specific connection properties

Returns:

the new XSManagedConnection instance

Throws:

`ResourceException`

See Also:

`ManagedConnectionFactory.createManagedConnection(Subject, ConnectionRequestInfo)`

---

getLogWriter

```
public PrintWriter getLogWriter()
                           throws ResourceException
```

Get the log writer for this XSManagedConnectionFactory instance.

Specified by:

`getLogWriter` in interface `ManagedConnectionFactory`

Returns:

PrintWriter

Throws:

`ResourceException`

See Also:

`ManagedConnectionFactory.getLogWriter()`

---

matchManagedConnections

```
public ManagedConnection matchManagedConnections(Set connectionSet,
                                                 Subject subject,
                                                 ConnectionRequestInfo cxRequestInfo)
                                          throws ResourceException
```

Returns a matched connection from the candidate set of connections.

Specified by:

`matchManagedConnections` in interface `ManagedConnectionFactory`

Parameters:

`connectionSet` - - candidate connection set

`subject` - - caller's security information

`cxRequestInfo` - - XS specific connection properties

Returns:

XSManagedConnection instance if acceptable match found otherwise null

Throws:

ResourceException

See Also:

ManagedConnectionFactory.matchManagedConnections(Set, Subject, ConnectionRequestInfo)

---

setLogWriter

```
public void setLogWriter(PrintWriter out)
                  throws ResourceException
```

Set the log writer for this ManagedConnectionFactory instance

Specified by:

setLogWriter in interface ManagedConnectionFactory

Parameters:

`out` - - PrintWriter - an out stream for error logging and tracing

Throws:

ResourceException

See Also:

ManagedConnectionFactory.setLogWriter(PrintWriter)

---

destroy

```
public void destroy()
```

Destroy this MCF and release any other resources.

---

getConnectionName

```
public String getConnectionName()
```

Get the eXtreme Scale specific property: ConnectionName

Returns:

The name of the eXtreme Scale client connection

---

setConnectionName

```
public void setConnectionName(String connName)
```

Set the eXtreme Scale specific property: ConnectionName

Parameters:

`connName` - - The name of the eXtreme Scale client connection

---

getObjectGridName

```
public String getObjectGridName()
```

Get the eXtreme Scale specific property: ObjectGridName

Returns:
The data grid name

---

setObjectGridName

`public void` **`setObjectGridName`**`(`[String](#) `objectGridName)`

Set the eXtreme Scale specific property: ObjectGrid Name

Parameters:
`objectGridName` - - The data grid name

---

getCatalogServiceEndpoints

`public` [String](#) **`getCatalogServiceEndpoints`**`()`

Get the eXtreme Scale specific property: CatalogServiceEndpoints

Returns:
The catalog service domain end points

---

setCatalogServiceEndpoints

`public void` **`setCatalogServiceEndpoints`**`(`[String](#) `catalogServiceEndpoints)`

Set the eXtreme Scale specific property: CatalogServiceEndpoints

Parameters:
`catalogServiceEndpoints` - - The catalog service domain end points

---

getCatalogServiceDomain

`public` [String](#) **`getCatalogServiceDomain`**`()`

Get the eXtreme Scale specific property: CatalogDomain

Returns:
The catalog service domain name defined in WebSphere Application Server

---

setCatalogServiceDomain

`public void` **`setCatalogServiceDomain`**`(`[String](#) `catalogServiceDomain)`

Set the eXtreme Scale specific property: CatalogDomain

Parameters:
`catalogServiceDomain` - - The catalog service domain name defined in WebSphere Application Server

---

getObjectGridURL

`public` [String](#) **`getObjectGridURL`**`()`

Get the eXtreme Scale specific property: ObjectGridURL

Returns:
The URL of the client data grid override XML file

setObjectGridURL

public void **setObjectGridURL**(String objectGridURL)

Set the eXtreme Scale specific property: ObjectGridURL

Parameters:

objectGridURL - The URL of the client data grid override XML file

---

getObjectGridResource

public String **getObjectGridResource**()

Get the eXtreme Scale specific property: ObjectGridResource

Returns:

The resource path of the client data grid override XML file

---

setObjectGridResource

public void **setObjectGridResource**(String objectGridResource)

Set the eXtreme Scale specific property: ObjectGridResource

Parameters:

objectGridResource - - The resource path of the client data grid override XML file

---

getClientPropertiesURL

public String **getClientPropertiesURL**()

Get the eXtreme Scale specific property: ClientPropertiesURL

Returns:

The URL of the client properties file

---

setClientPropertiesURL

public void **setClientPropertiesURL**(String clientPropsURL)

Set the eXtreme Scale specific property: ClientPropertiesURL

Parameters:

clientPropsURL - - The URL of the client properties file

---

getClientPropertiesResource

public String **getClientPropertiesResource**()

Get the eXtreme Scale specific property: ClientPropertiesResource

Returns:

The resource path of the client properties file

---

setClientPropertiesResource

public void **setClientPropertiesResource**(String clientPropsResource)

Get the eXtreme Scale specific property: ClientPropertiesResource

Parameters:

`clientPropsResource` - - The resource path of the client properties file

---

setLocalGrid

public void **setLocalGrid**(boolean useLocal)

Set the eXtreme Scale specific property: LocalGrid

When set to true, the application uses the XSConnectionSpec.setLocalObjectGrid(com.ibm.websphere.objectgrid.ObjectGrid) to set the ObjectGrid instance to a local, in-memory grid instance or shard instance. If set to false (the default), the connection will be configured as a client connection to a remote data grid.

Parameters:

`useLocal` - set to true, to disable normal client ObjectGrid connection management.

Since:

8.6, XC10 2.5

---

isLocalGrid

public boolean **isLocalGrid**()

Retrieve the eXtreme Scale specific property: LocalGrid

Returns:

answers true if the connection is to be used with a local, im-memory ObjectGrid instance. Answers false if the connection represents a normal client ObjectGrid connection.

Since:

8.6, XC10 2.5

---

hashCode

public int **hashCode**()

Specified by:

`hashCode` in interface `ManagedConnectionFactory`

Overrides:

`hashCode` in class `Object`

See Also:

ManagedConnectionFactory.hashCode()

---

equals

public boolean **equals**(Object obj)

Specified by:

`equals` in interface `ManagedConnectionFactory`

Overrides:

`equals` in class `Object`

See Also:

ManagedConnectionFactory.equals(Object)

---

toString

public String **toString**()

Overrides:

toString in class Object

---

com.ibm.websphere.xs.ra

Class XSConnectionSpec

java.lang.Object
  com.ibm.websphere.xs.ra.XSConnectionSpec

All Implemented Interfaces:
Cloneable, ConnectionSpec

```
public final class XSConnectionSpec
extends Object
implements ConnectionSpec, Cloneable
```

The ConnectionSpec for retrieving XSConnections from an XSConnectionFactory

Since:
8.5, XC10

See Also:
XSConnectionFactory

Constructor Summary

XSConnectionSpec() Creates a default XSCOnnectionSpec instance.

Method Summary

Object	**clone**()
boolean	**equals**(Object obj)
CredentialGenerator	**getCredentialGenerator**() Returns the CredentialGenerator object for this client security for the connection.

[erator](#)	
[Object](#)[Grid](#)	**getLocalObjectGrid**() Retrieve the local object grid instance used for the connection, if set.
[String](#)	**getObjectGridName**() Returns the object grid name for the connection.
int	**hashCode**()
boolean	**isBeginNoWriteThrough**() Returns flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.
boolean	**isMultiPartitionSupportEnabled**() Answers true if the [Session](#) will have the [Session.getTxCommitProtocol()](#) set to [Session.TxCommitProtocol.TWOPHASE](#).
void	**setBeginNoWriteThrough**(boolean beginNoWriteThrough) Set the boolean flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.
void	**setCredentialGenerator**([CredentialGenerator](#) credGen) Sets the `CredentialGenerator` object for the client security for the connection.
void	**setLocalObjectGrid**([ObjectGrid](#) instance) Sets a local ObjectGrid instance to be used for connections.
void	**setMultiPartitionSupportEnabled**(boolean mptEnabled) Set to true to retrieve a connection with a [Session](#) that has the [Session.getTxCommitProtocol()](#) set to [Session.TxCommitProtocol.TWOPHASE](#).
void	**setObjectGridName**([String](#) objectGridName) Sets the object grid name to be used for the connection.
[St](#)	

[r i n g](#)	**toString**()

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

XSConnectionSpec

public **XSConnectionSpec**()

Creates a default XSCOnnectionSpec instance.

Method Detail

isBeginNoWriteThrough

public boolean **isBeginNoWriteThrough**()

Returns flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.

Returns:

flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.

---

setBeginNoWriteThrough

public void **setBeginNoWriteThrough**(boolean beginNoWriteThrough)

Set the boolean flag indicating whether the Session transaction should be started using using the `Session.beginNoWriteThrough()` method.

Parameters:

`beginNoWriteThrough` -

See Also:

Session.begin(), Session.beginNoWriteThrough()

---

getObjectGridName

public String **getObjectGridName**()

Returns the object grid name for the connection.

Returns:

object grid name

---

setObjectGridName

public void **setObjectGridName**(String objectGridName)

Sets the object grid name to be used for the connection.

Parameters:

`objectGridName` - object grid name to be used for the connection

---

getLocalObjectGrid

`public ObjectGrid **getLocalObjectGrid**()`

Retrieve the local object grid instance used for the connection, if set.

Returns:

null if the connection retrieves its ObjectGrid instance normally from the physical connection, or non-null if `setLocalObjectGrid(ObjectGrid)` was specified.

Since:

8.6, XC10 2.5

---

setLocalObjectGrid

`public void **setLocalObjectGrid**(ObjectGrid instance)`

Sets a local ObjectGrid instance to be used for connections. The local ObjectGrid instance overrides the standard behavior and allows a physical connection to be used with a specific local ObjectGrid instance, such as a shard.

Parameters:

`instance` - the local ObjectGrid instance.

Since:

8.6, XC10 2.5

---

getCredentialGenerator

`public CredentialGenerator **getCredentialGenerator**()`

Returns the `CredentialGenerator` object for this client security for the connection.

Returns:

the argument that was passed to the `setCredGen(CredentialGenerator)` method of this object or `null` if `setCredGen` was not previously called for this object.

---

setCredentialGenerator

`public void **setCredentialGenerator**(CredentialGenerator credGen)`

Sets the `CredentialGenerator` object for the client security for the connection.

Parameters:

`credGen` - a CredentialGenerator object

See Also:

`CredentialGenerator`

---

setMultiPartitionSupportEnabled

`public void **setMultiPartitionSupportEnabled**(boolean mptEnabled)`

Set to true to retrieve a connection with a `Session` that has the `Session.getTxCommitProtocol()` set to `Session.TxCommitProtocol.TWOPHASE`.

Parameters:

mptEnabled - set to true if the Session should be configured to write to multiple partitions.

Since:

8.6, XC10 2.5

---

isMultiPartitionSupportEnabled

`public boolean` **`isMultiPartitionSupportEnabled`**`()`

Answers true if the `Session` will have the `Session.getTxCommitProtocol()` set to `Session.TxCommitProtocol.TWOPHASE`.

Returns:

true if the Session is capable of writing to multiple partitions.

Since:

8.6, XC10 2.5

---

clone

`public Object` **`clone`**`()`

Overrides:

`clone` in class `Object`

---

hashCode

`public int` **`hashCode`**`()`

Overrides:

`hashCode` in class `Object`

---

equals

`public boolean` **`equals`**`(Object obj)`

Overrides:

`equals` in class `Object`

---

toString

`public String` **`toString`**`()`

Overrides:

`toString` in class `Object`

---

com.ibm.websphere.xs.ra

Class XSResourceAdapter

java.lang.Object
  com.ibm.websphere.xs.ra.XSResourceAdapter

All Implemented Interfaces:

ResourceAdapter

```
public final class XSResourceAdapter
extends Object
implements ResourceAdapter
```

The eXtreme Scale resource adapter

Since:

8.5, XC10

Constructor Summary

XSResourceAdapter()

Method Summary

void	**endpointActivation**(MessageEndpointFactory msgEndpointFactory, ActivationSpec actSpec) This method does nothing as endpoint activation is not supported
void	**endpointDeactivation**(MessageEndpointFactory msgEndpointFactory, ActivationSpec spec) This method does nothing as endpoint deactivation is not supported
XAResource[]	**getXAResources**(ActivationSpec[] spec) This method returns null as the XA protocol is not supported
void	**start**(BootstrapContext ctx) This method initializes the eXtreme Scale resource adapter instance
void	**stop**()

id	This method performs the shutdown of the eXtreme Scale resource adapter instance
String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XSResourceAdapter

```
public XSResourceAdapter()
```

Method Detail

endpointActivation

```
public void endpointActivation(MessageEndpointFactory msgEndpointFactory,
                               ActivationSpec actSpec)
                        throws ResourceException
```

This method does nothing as endpoint activation is not supported

Specified by:
endpointActivation in interface ResourceAdapter

Throws:
ResourceException

See Also:
ResourceAdapter.endpointActivation(MessageEndpointFactory,ActivationSpec)

---

endpointDeactivation

```
public void endpointDeactivation(MessageEndpointFactory msgEndpointFactory,
                                 ActivationSpec spec)
```

This method does nothing as endpoint deactivation is not supported

Specified by:
endpointDeactivation in interface ResourceAdapter

Throws:
ResourceException

See Also:
ResourceAdapter.endpointDeactivation(MessageEndpointFactory,ActivationSpec)

---

getXAResources

```
public XAResource[] getXAResources(ActivationSpec[] spec)
                            throws ResourceException
```

This method returns null as the XA protocol is not supported

Specified by:
getXAResources in interface ResourceAdapter

Throws:
ResourceException

See Also:
ResourceAdapter.getXAResources(ActivationSpec[])

---

start

```
public void start(BootstrapContext ctx)
          throws ResourceAdapterInternalException
```

This method initializes the eXtreme Scale resource adapter instance

Specified by:
start in interface ResourceAdapter

Throws:
ResourceAdapterInternalException

See Also:
ResourceAdapter.start(BootstrapContext)

---

stop

```
public void stop()
```

This method performs the shutdown of the eXtreme Scale resource adapter instance

Specified by:
stop in interface ResourceAdapter

See Also:
ResourceAdapter.stop()

---

toString

```
public String toString()
```

Overrides:
toString in class Object

---

com.ibm.websphere.xs.ra

Class XSConnection

java.lang.Object
  com.ibm.websphere.xs.ra.XSConnection

All Implemented Interfaces:
Connection

```
public final class XSConnection
extends Object
implements Connection
```

An XSConnection represents an application level connection handle to an eXtreme Scale ObjectGrid.

Since:
8.5, XC10

Constructor Summary

XSConnection()

Method Summary

void	**close**() Initiates close of the connection handle at the application level.
Interaction	**createInteraction**() Creates an Interaction associated with this Connection.
LocalTransaction	**getLocalTransaction**() Returns an LocalTransaction instance that enables a component to demarcate local transactions on the Connection.

[tion](#)	
[ConnectionMetaData](#)	**[getMetaData](#)**() Gets the information on the underlying eXtreme Scale instance represented through an active connection.
[ResultSetInfo](#)	**[getResultSetInfo](#)**() Gets the information on the ResultSet functionality supported by a connected eXtreme Scale instance.
[Session](#)	**[getSession](#)**() Retrieve the ObjectGrid Session associated with this connection.
[String](#)	**[toString](#)**()

Methods inherited from class java.lang.[Object](#)
[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Constructor Detail

XSConnection

public **XSConnection**()

Method Detail

getSession

```
public Session getSession()
                  throws ResourceException
```

Retrieve the ObjectGrid Session associated with this connection.

Returns:
The ObjectGrid Session
Throws:
ResourceException

---

close

```
public void close()
                  throws ResourceException
```

Initiates close of the connection handle at the application level.

Specified by:
close in interface Connection

Throws:
ResourceException
See Also:
Connection.close()

---

createInteraction

```
public Interaction createInteraction()
                           throws ResourceException
```

Creates an Interaction associated with this Connection.

Specified by:
createInteraction in interface Connection

Returns:
Interaction instance
Throws:
ResourceException
NotSupportedException
See Also:
Connection.createInteraction()

---

getLocalTransaction

```
public LocalTransaction getLocalTransaction()
                                    throws ResourceException
```

Returns an LocalTransaction instance that enables a component to demarcate local transactions on the Connection.

Specified by:
getLocalTransaction in interface Connection

Returns:
LocalTransaction instance
Throws:
ResourceException
See Also:
Connection.getLocalTransaction()

---

getMetaData

```
public ConnectionMetaData getMetaData()
                            throws ResourceException
```

Gets the information on the underlying eXtreme Scale instance represented through an active connection.

Specified by:
getMetaData in interface Connection

Returns:
WXS ConnectionMetaData

Throws:
ResourceException

See Also:
Connection.getMetaData()

---

getResultSetInfo

```
public ResultSetInfo getResultSetInfo()
                              throws ResourceException
```

Gets the information on the ResultSet functionality supported by a connected eXtreme Scale instance.

Specified by:
getResultSetInfo in interface Connection

Returns:
ResultSetInfo instance

Throws:
ResourceException

See Also:
Connection.getResultSetInfo()

---

toString

```
public String toString()
```

Overrides:
toString in class Object

---

com.ibm.websphere.xs.ra

Class XSConnectionFactory

java.lang.Object
  com.ibm.websphere.xs.ra.XSConnectionFactory

All Implemented Interfaces:
Serializable, Referenceable, ConnectionFactory

```
public final class XSConnectionFactory
extends Object
implements ConnectionFactory
```

The XSConnectionFactory creates connections to eXtreme Scale ObjectGrids.

Usage example:

```
InitialContext ctx = new InitialContext();
XSConnectionFactory cf = (XSConnectionFactory) ctx.lookup("java:comp/env/wxsconnection");
XSConnection con = cf.getConnection("MyGrid");
Session ogSession = con.getSession();
...
con.close();
```

For additional examples, see the package documentation.

Since:
8.5, XC10

See Also:
Serialized Form

Constructor Summary

XSConnectionFactory()

Method Summary

String	**getCatalogServiceDomain**() Get the eXtreme Scale specific property: CatalogDomain
String	**getCatalogServiceEndpoints**() Get the eXtreme Scale specific property: CatalogServiceEndpoints

g	
String	**getClientPropertiesResource**() Get the eXtreme Scale specific property: ClientPropertiesResource
String	**getClientPropertiesURL**() Get the eXtreme Scale specific property: ClientPropertiesURL
Connection	**getConnection**() Retrieve an XSConnection for the ObjectGrid configured on the ConnectionFactory.
Connection	**getConnection**(ConnectionSpec spec) Retrieve an XSConnection for the ObjectGrid using the specified XSConnectionSpec properties.
Connection	**getConnection**(String gridName) Retrieve an XSConnection for the specified ObjectGrid name.
String	**getConnectionName**() Get the eXtreme Scale specific property: ConnectionName
ResourceAdapterMet	**getMetaData**() Retrieve the metadata information regarding the eXtreme Scale resource adapter

[aData](#)	
[String](#)	**getObjectGridName**() Get the eXtreme Scale specific property: ObjectGridName
[String](#)	**getObjectGridResource**() Get the eXtreme Scale specific property: ObjectGridResource
[String](#)	**getObjectGridURL**() Get the eXtreme Scale specific property: ObjectGridURL
[RecordFactory](#)	**getRecordFactory**() Retrieve the RecordFactory instance
[Reference](#)	**getReference**() Gets the Reference instance
boolean	**isLocalGrid**() Get the eXtreme Scale specific property: LocalGrid
void	**setReference**([Reference](#) ref) Sets the Reference instance
[String](#)	**toString**()

Methods inherited from class java.lang.[Object](#)

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XSConnectionFactory

public **XSConnectionFactory**()

Method Detail

setReference

public void **setReference**(Reference ref)

Sets the Reference instance

Parameters:
ref - - Reference instance
See Also:
Referenceable.setReference(Reference)

---

getReference

public Reference **getReference**()
throws NamingException

Gets the Reference instance

Specified by:
getReference in interface Referenceable

Returns:
Reference instance
Throws:
NamingException
See Also:
Referenceable.getReference()

---

getConnection

public Connection **getConnection**()
throws ResourceException

Retrieve an XSConnection for the ObjectGrid configured on the ConnectionFactory.

Specified by:
getConnection in interface ConnectionFactory

Returns:
the connection to the eXtreme Scale ObjectGrid
Throws:
ResourceException
See Also:
ConnectionFactory.getConnection()

---

getConnection

```
public Connection getConnection(ConnectionSpec spec)
                         throws ResourceException
```

Retrieve an XSConnection for the ObjectGrid using the specified XSConnectionSpec properties.

Specified by:

getConnection in interface ConnectionFactory

Parameters:

spec - the XSConnectionSpec properties used to retrieve the correct connection.

Returns:

the connection to the eXtreme Scale ObjectGrid

Throws:

ResourceException

See Also:

ConnectionFactory.getConnection(ConnectionSpec), XSConnectionSpec

---

getConnection

```
public Connection getConnection(String gridName)
                         throws ResourceException
```

Retrieve an XSConnection for the specified ObjectGrid name.

Parameters:

gridName - - The ObjectGrid name

Returns:

the connection to the eXtreme Scale ObjectGrid

Throws:

ResourceException

---

getMetaData

```
public ResourceAdapterMetaData getMetaData()
                                  throws ResourceException
```

Retrieve the metadata information regarding the eXtreme Scale resource adapter

Specified by:

getMetaData in interface ConnectionFactory

Returns:

The eXtreme Scale resource adapter metadata

Throws:

ResourceException

See Also:

ConnectionFactory.getMetaData()

---

getRecordFactory

```
public RecordFactory getRecordFactory()
                               throws ResourceException
```

Retrieve the RecordFactory instance

Specified by:

getRecordFactory in interface ConnectionFactory

Throws:

NotSupportedException
ResourceException

See Also:

ConnectionFactory.getRecordFactory()

---

toString

public String **toString**()

Overrides:

toString in class Object

---

getCatalogServiceDomain

public String **getCatalogServiceDomain**()

Get the eXtreme Scale specific property: CatalogDomain

Returns:

The catalog service domain name defined in WebSphere Application Server

---

getCatalogServiceEndpoints

public String **getCatalogServiceEndpoints**()

Get the eXtreme Scale specific property: CatalogServiceEndpoints

Returns:

The catalog service domain end points

---

getClientPropertiesResource

public String **getClientPropertiesResource**()

Get the eXtreme Scale specific property: ClientPropertiesResource

Returns:

The resource path of the client properties file

---

getClientPropertiesURL

public String **getClientPropertiesURL**()

Get the eXtreme Scale specific property: ClientPropertiesURL

Returns:

The URL of the client properties file

---

getConnectionName

public String **getConnectionName**()

Get the eXtreme Scale specific property: ConnectionName

Returns:

The name of the eXtreme Scale client connection

---

getObjectGridName

public String **getObjectGridName**()

Get the eXtreme Scale specific property: ObjectGridName

Returns:
The data grid name

---

getObjectGridResource

public String **getObjectGridResource**()

Get the eXtreme Scale specific property: ObjectGridResource

Returns:
The resource path of the client data grid override XML file

---

getObjectGridURL

public String **getObjectGridURL**()

Get the eXtreme Scale specific property: ObjectGridURL

Returns:
The URL of the client data grid override XML file

---

isLocalGrid

public boolean **isLocalGrid**()

Get the eXtreme Scale specific property: LocalGrid

Returns:
Is this managed connection factory used only for access to a local ObjectGrid instance?

---

com.ibm.websphere.xs.ra

Interface ObjectGridJ2CConnectionMBean

public interface **ObjectGridJ2CConnectionMBean**

Allows administration clients control the lifecycle of the eXtreme Scale resource adapter's connection.

The object name pattern for this MBean is:

```
 com.ibm.websphere.objectgrid:type=ObjectGridJ2CConnection,objectGridName=<objectgrid>,domain=<dom
ain name>,connectionName=<connection name>
```

Note: Additional properties may be included.

Since:

8.5, XC10

Field Summary

static String	**CONNECTIONNAME_DEFAULT** The default connection name prefix when not specified.
static String	**CONNECTIONSTATUS_CONNECTED** The connection status indicating that a connection has been established.
static String	**CONNECTIONSTATUS_DISCONNECTED** The connection status indicating that client is no longer connected.
s	

static String	**MBEAN_TYPE** The MBean type

Method Summary

String	**getCatalogServiceEndpoints**() The catalog service endpoints of the catalog service.
String	**getConnectionName**() The name of the connection as specified on the J2C ConnectionFactory, or "DEFAULT" if not specified.
String	**getConnectionStatus**() The status of the connection.
String	**getDomainName**() The domain name of the catalog service domain as reported by the catalog service.
String	**getObjectGridName**() The name of the ObjectGrid that is connected.
void	**resetObjectGridConnection**() Reset the ObjectGrid connection.

Field Detail

MBEAN_TYPE

```
static final String MBEAN_TYPE
```

The MBean type

See Also:

Constant Field Values

CONNECTIONSTATUS_CONNECTED

static final String **CONNECTIONSTATUS_CONNECTED**

The connection status indicating that a connection has been established.

See Also:
Constant Field Values

CONNECTIONSTATUS_DISCONNECTED

static final String **CONNECTIONSTATUS_DISCONNECTED**

The connection status indicating that client is no longer connected.

See Also:
Constant Field Values

CONNECTIONNAME_DEFAULT

static final String **CONNECTIONNAME_DEFAULT**

The default connection name prefix when not specified.

See Also:
Constant Field Values

Method Detail

resetObjectGridConnection

void **resetObjectGridConnection**()

Reset the ObjectGrid connection.

This destroys the client connection to the ObjectGrid, including any local cache that may be created.

Subsequent uses of the ManagedConnectionFactory will result in a new ObjectGrid connection.

---

getObjectGridName

String **getObjectGridName**()

The name of the ObjectGrid that is connected.

Returns:
the ObjectGrid name

---

getConnectionName

String **getConnectionName**()

The name of the connection as specified on the J2C ConnectionFactory, or "DEFAULT" if not specified. If there are multiple connections with the same attributes in the same process, the "DEFAULT" name will have an integer appended.

Returns:
the connection name.

---

getDomainName

`String` **`getDomainName`**`()`

The domain name of the catalog service domain as reported by the catalog service.

Returns:
the catalog service domain name.

---

getCatalogServiceEndpoints

`String` **`getCatalogServiceEndpoints`**`()`

The catalog service endpoints of the catalog service.

Returns:
the catalog service endpoints

---

getConnectionStatus

`String` **`getConnectionStatus`**`()`

The status of the connection. Valid states include "CONNECTED" or "DISCONNECTED".

Other states may be returned in the future.

Returns:
one of the CONNECTIONSTATUS constants defined in this interface.

Package com.ibm.websphere.xsa

Class Summary	
RestValue	**Deprecated.** *Use the `RestValue` interface and the `RestValueFactory`.*

Overview | Package | Class | Tree | Serialized | Deprecated | Index | Help

PREV PACKAGE NEXT PACKAGE | FRAMES | NO FRAMES | All Classes

IBM WebSphere® DataPower® XC10 Appliance Release 2.5 Client API Specification

com.ibm.websphere.xsa

Class RestValue

java.lang.Object
  com.ibm.websphere.xsa.RestValue

All Implemented Interfaces:
com.ibm.websphere.xs.rest.RestValue, Externalizable, Serializable

---

Deprecated. *Use the `RestValue` interface and the `RestValueFactory`.*

```
public class RestValue
extends Object
implements com.ibm.websphere.xs.rest.RestValue, Externalizable
```

This object is used in the XC10 REST Gateway. It is a data wrapper for data to be accessed via XC10 REST Gateway. With its wrapped data, it will be stored in an object grid's map entry, with a String as the key of the map entry.

Since:
8.6, XC10 1.0.0.4
See Also:
Serialized Form

Field Summary

protected boolean	**compressed** **Deprecated.** Flag to indicate if the value is compressed or not
protected String	**contentType** **Deprecated.** Content type of the contained value

g	
protected byte[]	**value** **Deprecated.** Contained value

Constructor Summary

RestValue() **Deprecated.** Constructor
RestValue(byte[] value, String contentType) **Deprecated.** Constructor

Method Summary

boolean	**equals**(Object obj) **Deprecated.**
String	**getContentType**() **Deprecated.** Return content type of the contained value
byte[]	**getValue**() **Deprecated.** Return contained value
int	**hashCode**() **Deprecated.**
boolean	**isCompressed**() **Deprecated.** Indicate if the contained value is compressed
void	**readExternal**(ObjectInput in) **Deprecated.**
vo	**setCompressed**(boolean compressed)

void	**Deprecated.** Set indicator to indicate if the contained value is compressed
void	**setContentType**(String contentType) **Deprecated.** Set content type of the contained value
void	**setValue**(byte[] value) **Deprecated.** Set contained value
String	**toString**() **Deprecated.**
void	**writeExternal**(ObjectOutput out) **Deprecated.**

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

value

protected byte[] **value**

Deprecated.
Contained value

---

contentType

protected String **contentType**

Deprecated.
Content type of the contained value

---

compressed

protected boolean **compressed**

Deprecated.
Flag to indicate if the value is compressed or not

Constructor Detail

RestValue

public **RestValue**()

Deprecated.

Constructor

Since:
8.6, XC10 1.0.0.4

RestValue

```
public RestValue(byte[] value,
                 String contentType)
```

Deprecated.
Constructor

Parameters:
`value` - A value to be wrapped by RestValue class.
`contentType` - Content type of the value.

Since:
8.6, XC10 1.0.0.4

Method Detail

hashCode

```
public int hashCode()
```

Deprecated.

Overrides:
hashCode in class Object

---

equals

```
public boolean equals(Object obj)
```

Deprecated.

Overrides:
equals in class Object

---

toString

```
public String toString()
```

Deprecated.

Overrides:
toString in class Object

---

getValue

```
public byte[] getValue()
```

Deprecated.
Return contained value

Specified by:
`getValue` in interface `com.ibm.websphere.xs.rest.RestValue`

Returns:
An array of byte that represents the contained value
Since:
8.6, XC10 1.0.0.4

---

setValue

```
public void setValue(byte[] value)
```

Deprecated.
Set contained value

Parameters:
`value` - An array of byte that represents the contained value
Since:
8.6, XC10 1.0.0.4

---

getContentType

```
public String getContentType()
```

Deprecated.
Return content type of the contained value

Specified by:
`getContentType` in interface `com.ibm.websphere.xs.rest.RestValue`

Returns:
A content type
Since:
8.6, XC10 1.0.0.4

---

setContentType

```
public void setContentType(String contentType)
```

Deprecated.
Set content type of the contained value

Parameters:
`contentType` - A content type
Since:
8.6, XC10 1.0.0.4

---

readExternal

```
public void readExternal(ObjectInput in)
                  throws IOException,
                         ClassNotFoundException
```

Deprecated.

Specified by:
`readExternal` in interface `Externalizable`

Throws:
`IOException`
`ClassNotFoundException`

writeExternal

```
public void writeExternal(ObjectOutput out)
                   throws IOException
```

Deprecated.

Specified by:

writeExternal in interface Externalizable

Throws:

IOException

---

isCompressed

```
public boolean isCompressed()
```

Deprecated.
Indicate if the contained value is compressed

Returns:

true if the contained value is compressed

Since:

8.6, XC10 1.0.0.4

---

setCompressed

```
public void setCompressed(boolean compressed)
```

Deprecated.
Set indicator to indicate if the contained value is compressed

Parameters:

`compressed` - A boolean value that indicate if the contained value is compressed

---

IBM WebSphere eXtreme Scale Client for .NET API Specification

Contents

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Namespaces

Namespace	Description
	The IBM.WebSphere.Caching namespace is the parent namespace for the the WebSphere eXtreme Scale C# client data access application programming interfaces. The primary entry point for interacting with the data grid is the GridManagerFactory, which provides access to the IGridManager and IGrid interfaces.
IBM.WebSphere.Caching	The IGridManager interface provides methods to connect and disconnect from a catalog service domain. It also provides access to IGrid instances from the catalog service domain connections. The IGrid interface allows clients to interact with a named data grid using various map interfaces, such as the IGridMapPessimisticTx TKey, TValue and IGridMapPessimisticAutoTx TKey, TValue . For additional information about data access APIs, see the IBM.WebSphere.Caching.Map namespace documentation.
IBM.WebSphere.Caching.Map	The IBM.WebSphere.Caching.Map namespace includes the data access application programming interfaces. See the IBM.WebSphere.Caching namespace documentation for a description on how to access a map. The eXtreme Scale client supports transactional data access to individual maps using automatic and manual transactions. The following maps are available: • The IGridMapPessimisticAutoTx TKey, TValue interface provides automatic transactions. • The IGridMapPessimisticTx TKey, TValue interface provides manual transaction demarcation. See each respective interface for programming examples.
IBM.WebSphere.Caching.Security	The IBM.WebSphere.Caching.Security namespace includes the application programming interfaces specific to security. The eXtreme Scale client supports transport security and authenticiation and authorization using the ICredentialGenerator interface. Security is configured using a client properties file.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IBM.WebSphere.Caching namespace is the parent namespace for the the WebSphere eXtreme Scale C# client data access application programming interfaces.

The primary entry point for interacting with the data grid is the GridManagerFactory, which provides access to the IGridManager and IGrid interfaces.

The IGridManager interface provides methods to connect and disconnect from a catalog service domain. It also provides access to IGrid instances from the catalog service domain connections.

The IGrid interface allows clients to interact with a named data grid using various map interfaces, such as the IGridMapPessimisticTx TKey, TValue and IGridMapPessimisticAutoTx TKey, TValue .

For additional information about data access APIs, see the IBM.WebSphere.Caching.Map namespace documentation.

Classes

Class	Description
AvailabilityException	An AvailabilityException exception occurs when a target is not in the correct state to handle a request that it receives.
ClientServerTransactionCallbackException	A ClientServerTransactionCallbackException exception occurs when a method call to the client/server TransactionCallback encounters a remote request problem.
GridConfigurationException	An GridConfigurationException exception occurs when a configuration problem is found. This exception might occur when the configuration specified in the deployment policy, ObjectGrid descriptor, or security descriptor is not correct.
GridException	An GridException exception is the base exception class for all checked exceptions that occur in the product.
GridManagerFactory	The GridManagerFactory is a factory for IGridManager instances, and is the entrypoint for all interactions with the data grid.
GridServerRuntimeException	A GridServerRuntimeException exception is a generic wrapper for exceptions that occur in the server runtime.
LifecycleFailedException	A LifecycleFailedException exception occurs on unexpected lifecycle states.
MixedTransportException	A MixedTransportException is thrown when server and client have mismatched transport. For example, a server is using ORB, but the client is eXtremeIO
NoActiveTransactionException	A NoActiveTransactionException exception indicates that no active transactions exist.
NotReentrantException	A NotReentrantException occurs when a thread tries to run a map operation, such as calling a method on ObjectMap interface, when another thread is already running a map operation for the Session. A Session object can be used by a single thread only to perform concurrent map operations.
OrderedDictionary TKey, TValue	A generic version of the non-generic OrderedDictionary class.
OrderedDictionary TKey,	

TValue_Enumerator	The enumerator for interating through the OrderedDictionary TKey, TValue .
ReplicationVotedToRollbackTransactionException	A ReplicationVotedToRollbackTransactionException exception occurs when a transaction was rolled back because some or all of the synchronous replicas did not apply the transaction.
TransactionAlreadyActiveException	A TransactionAlreadyActiveException exception occurs to indicate that a transaction is already active for the current Session. This exception does not cause the current active transaction to be rolled back, so the isTransactionActive method returns true.
TransactionCallbackException	A TransactionCallbackException exception occurs when a TransactionCallback method call fails.
TransactionException	A TransactionException exception is a general locking exception that indicates something went wrong with a transaction. Use the isTransactionActive() and wasTransactionRolledBack() methods to determine whether transaction is still active or was rolled back as a result of this exception.
TransactionTimeoutException	A TransactionTimeoutException exception occurs when a transaction exceeds the transaction timeout value that was specified on the ObjectGrid or Session.

Interfaces

Interface	Description
ICatalogDomainInfo	Identifies a catalog service domain to be used for connecting to an eXtreme Scale data grid.
ICatalogDomainManager	The ICatalogDomainManager is a factory for ICatalogDomainInfo objects used to connect to a catalog service domain. Use the CatalogDomainManager to retrieve an ICatalogDomainMananager instance.
IClientConnectionContext	The handle to a connection to a catalog service domain. An IClientConnectionContext is returned from the Connect(ICatalogDomainInfo) method when connecting to a data grid. When finished with the data grid, use the Disconnect(IClientConnectionContext) method to disconnect from the data grid.
IGrid	An IGrid is the client's access to a named data grid and cooresponds to a configured ObjectGrid in the catalog service domain. An IGrid is thread safe and should be cached and reused by the application and is valid until the connection is severed. Use an IGrid to get map instances. Maps are defined on the data grid and have unique names within each configured ObjectGrid.
IGridManager	Provides methods for connecting to data grids. Use the GridManagerFactory to obtain an instance to an IGridManager instance.
IGridTransaction	Defines the interface for a transaction.
IOrderedDictionary TKey, TValue	Specifies a generic version of the non-generic IOrderedDictionary interface.
ITransactionable	Implementors of ITransactionable provide transaction demarcation semantics.

Enumerations

Enumer

ation	Description
AvailabilityState	Each shard in a distributed data has an associated availability state. This state determines if the shard can process incoming requests.
TxnIsolationLevel	Specifies an enumeration that defines the valid transaction isolation level values.

Remarks

Note: The client API in this namespace and child namespaces will evolve in future releases. Therefore, to avoid problems with incompatible bindings in the future, interfaces should not be directly implemented nor extended unless explicitly noted in the interface documentation.
Examples

The following example illustrates how to connect to a data grid, retrieve an entry from a map, and disconnect from the grid:
Copy to ClipboardPrint

```
// Retrieve the IGridManager instance. IGridManager gm = GridManagerFactory.GetGridManager(); // Id
```


Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An AvailabilityException exception occurs when a target is not in the correct state to handle a request that it receives.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching AvailabilityException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

AvailabilityException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AvailabilityException type exposes the following members.

Constructors

Name	Description
AvailabilityException	Initializes a new instance of the AvailabilityException class.
AvailabilityException(Exception)	Initializes a new instance of the AvailabilityException class with the specified exception cause.
AvailabilityException(String)	Initializes a new instance of the AvailabilityException class with the specified error message.
AvailabilityException(String, Exception)	Initializes a new instance of the AvailabilityException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	NewAvailabilityState	Gets or sets the availability state. This state controls if the shard can process incoming requests.
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

AvailabilityException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
AvailabilityException	Initializes a new instance of the AvailabilityException class.
AvailabilityException(Exception)	Initializes a new instance of the AvailabilityException class with the specified exception cause.
AvailabilityException(String)	Initializes a new instance of the AvailabilityException class with the specified error message.
AvailabilityException(String, Exception)	Initializes a new instance of the AvailabilityException class with the specified error message and exception cause.

Back to Top

See Also

AvailabilityException Class
AvailabilityException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that caused of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AvailabilityException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that caused of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

AvailabilityException Class
AvailabilityException Members
AvailabilityException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AvailabilityException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

AvailabilityException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AvailabilityException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
NewAvailabilityState	Gets or sets the availability state. This state controls if the shard can process incoming requests.
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

AvailabilityException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the availability state. This state controls if the shard can process incoming requests.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

AvailabilityException Class
AvailabilityException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Each shard in a distributed data has an associated availability state. This state determines if the shard can process incoming requests.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Members

Member name	Value	Description
OFFLINE	0	An AvailabilityState.OFFLINE state indicates that a shard is offline and cannot process requests.
PRELOAD	1	An AvailabilityState.PRELOAD state indicates that a shard is in the preload state. When a shard is in the preload state, the shard allows requests only from a client that is preloading data into the data grid. All other requests are rejected.
ONLINE	2	An AvailabilityState.ONLINE state indicates that a shard is online. A shard that is online is available for processing requests.
QUIESCE	3	An AvailabilityState.QUIESCE state indicates that a shard is in quiesce. Quiesce is a transitional state. Shards that are in the quiesce state are being taken offline.
UNKNOWN	4	An AvailabilityState.UNKNOWN state indicates that the availability state of the shard could not be determined.

See Also

IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ClientServerTransactionCallbackException exception occurs when a method call to the client/server TransactionCallback encounters a remote request problem.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionCallbackException
        IBM.WebSphere.Caching ClientServerTransactionCallbackException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ClientServerTransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerTransactionCallbackException type exposes the following members.

Constructors

Name	Description
ClientServerTransactionCallbackException	Initializes a new instance of the ClientServerTransactionCallbackException class.
ClientServerTransactionCallbackException(Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified exception cause.
ClientServerTransactionCallbackException(String)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message.
ClientServerTransactionCallbackException(String, Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ClientServerTransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ClientServerTransactionCallbackException	Initializes a new instance of the ClientServerTransactionCallbackException class.
ClientServerTransactionCallbackException(Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified exception cause.
ClientServerTransactionCallbackException(String)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message.
ClientServerTransactionCallbackException(String, Exception)	Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message and exception cause.

Back to Top

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerTransactionCallbackException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerTransactionCallbackException Class
ClientServerTransactionCallbackException Members
ClientServerTransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerTransactionCallbackException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerTransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerTransactionCallbackException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerTransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An GridConfigurationException exception occurs when a configuration problem is found. This exception might occur when the configuration specified in the deployment policy, ObjectGrid descriptor, or security descriptor is not correct.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching GridConfigurationException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridConfigurationException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridConfigurationException type exposes the following members.

Constructors

Name	Description
GridConfigurationException	Initializes a new instance of the GridConfigurationException class.
GridConfigurationException(Exception)	Initializes a new instance of the GridConfigurationException class with a specified exception cause.
GridConfigurationException(String)	Initializes a new instance of the GridConfigurationException class with the specified error message.
GridConfigurationException(String, Exception)	Initializes a new instance of the GridConfigurationException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)

	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridConfigurationException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridConfigurationException	Initializes a new instance of the GridConfigurationException class.
GridConfigurationException(Exception)	Initializes a new instance of the GridConfigurationException class with a specified exception cause.
GridConfigurationException(String)	Initializes a new instance of the GridConfigurationException class with the specified error message.
GridConfigurationException(String, Exception)	Initializes a new instance of the GridConfigurationException class with the specified error message and exception cause.

Back to Top

See Also

GridConfigurationException Class
GridConfigurationException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

[Send Feedback]

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridConfigurationException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridConfigurationException Class
GridConfigurationException Members
GridConfigurationException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

[Send Feedback]

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridConfigurationException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

GridConfigurationException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridConfigurationException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridConfigurationException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

An GridException exception is the base exception class for all checked exceptions that occur in the product.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching AvailabilityException
      IBM.WebSphere.Caching GridConfigurationException
      IBM.WebSphere.Caching LifecycleFailedException
      IBM.WebSphere.Caching.Map CacheKeyNotFoundException
      IBM.WebSphere.Caching.Map DuplicateKeyException
      IBM.WebSphere.Caching.Map LoaderException
      IBM.WebSphere.Caching.Map LockException
      IBM.WebSphere.Caching.Map LockStrategyNotSupportedException
      IBM.WebSphere.Caching.Map MultiplePartitionWriteException
      IBM.WebSphere.Caching.Map OptimisticCollisionException
      IBM.WebSphere.Caching.Map ReadOnlyException
      IBM.WebSphere.Caching.Map TargetNotAvailableException
      IBM.WebSphere.Caching.Map UndefinedMapException
      IBM.WebSphere.Caching MixedTransportException
      IBM.WebSphere.Caching NotReentrantException
      IBM.WebSphere.Caching TransactionCallbackException
      IBM.WebSphere.Caching TransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridException Members
IBM.WebSphere.Caching Namespace

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridException type exposes the following members.

Constructors

Name	Description
GridException	Initializes a new instance of the GridException class.
GridException(Exception)	Initializes a new instance of the GridException class with a specified exception cause.
GridException(String)	Initializes a new instance of the GridException class with the specified error message.
GridException(String, Exception)	Initializes a new instance of the GridException class with the specified error message and exception cause.

Back to Top

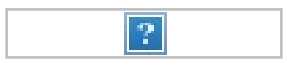
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

GridException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridException	Initializes a new instance of the GridException class.
GridException(Exception)	Initializes a new instance of the GridException class with a specified exception cause.
GridException(String)	Initializes a new instance of the GridException class with the specified error message.
GridException(String, Exception)	Initializes a new instance of the GridException class with the specified error message and exception cause.

Back to Top

See Also

GridException Class
GridException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridException Class
GridException Members
GridException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridException type exposes the following members.
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top
See Also

GridException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridException type exposes the following members.
Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

GridException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridManagerFactory is a factory for IGridManager instances, and is the entrypoint for all interactions with the data grid.

Inheritance Hierarchy

System Object
  IBM.WebSphere.Caching GridManagerFactory

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridManagerFactory Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridManagerFactory type exposes the following members.

Methods

	Name	Description
	GetGridManager	Retrieves the IGridManager singleton instance that will be used by the client process to connect with the ObjectGrid.

Back to Top

See Also

GridManagerFactory Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridManagerFactory type exposes the following members.

Methods

	Name	Description
	GetGridManager	Retrieves the IGridManager singleton instance that will be used by the client process to connect with the ObjectGrid.

Back to Top

See Also

GridManagerFactory Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the IGridManager singleton instance that will be used by the client process to connect with the ObjectGrid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Return Value

The IGridManager singleton

See Also

GridManagerFactory Class
GridManagerFactory Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A GridServerRuntimeException exception is a generic wrapper for exceptions that occur in the server runtime.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridServerRuntimeException
      IBM.WebSphere.Caching.Security AccessControlException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridServerRuntimeException type exposes the following members.

Constructors

Name	Description
GridServerRuntimeException(String)	Initializes a new instance of the GridServerRuntimeException class.
GridServerRuntimeException(String, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message and the Java exception class name.
GridServerRuntimeException(String, Exception, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message, a reference to the inner exception that is the cause of this exception, and the Java exception class name.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Provides a user-readable representation of this GridServerRuntimeException exception. (Overrides Exception ToString .)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	JavaException ClassName	Identifies the Java Exception class name that this exception identifies.
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

GridServerRuntimeException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridServerRuntimeException(String)	Initializes a new instance of the GridServerRuntimeException class.
GridServerRuntimeException(String, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message and the Java exception class name.
GridServerRuntimeException(String, Exception, String)	Initializes a new instance of the GridServerRuntimeException class with the specified error message, a reference to the inner exception that is the cause of this exception, and the Java exception class name.

Back to Top

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridServerRuntimeException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

javaExceptionClassName
Type: System String
Specifies the Java exception class name.

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
GridServerRuntimeException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridServerRuntimeException class with the specified error message and the Java exception class name.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

javaExceptionClassName
Type: System String
Specifies the Java exception class name.

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
GridServerRuntimeException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridServerRuntimeException class with the specified error message, a reference to the inner exception that is the cause of this exception, and the Java exception class name.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

innerException
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

javaExceptionClassName
Type: System String
Specifies the Java exception class name.

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
GridServerRuntimeException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridServerRuntimeException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Provides a user-readable representation of this GridServerRuntimeException exception. (Overrides Exception ToString .)

Back to Top

See Also

GridServerRuntimeException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Provides a user-readable representation of this GridServerRuntimeException exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

The user-readable representation of this exception.

Implements

_Exception ToString

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

The GridServerRuntimeException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
JavaExceptionClassName	Identifies the Java Exception class name that this exception identifies.
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridServerRuntimeException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Identifies the Java Exception class name that this exception identifies.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

GridServerRuntimeException Class
GridServerRuntimeException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Identifies a catalog service domain to be used for connecting to an eXtreme Scale data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ICatalogDomainInfo Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainInfo type exposes the following members.

Properties

Name	Description
CatalogServerAddresses	Retrieves the configured catalog service addresses. See the CreateCatalogDomainInfo(String) method for more details of the format of the catalog service address string.

Back to Top

See Also

ICatalogDomainInfo Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainInfo type exposes the following members.

Properties

Name	Description
CatalogServerAddresses	Retrieves the configured catalog service addresses. See the CreateCatalogDomainInfo(String) method for more details of the format of the catalog service address string.

Back to Top

See Also

ICatalogDomainInfo Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the configured catalog service addresses. See the CreateCatalogDomainInfo(String) method for more details of the format of the catalog service address string.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICatalogDomainInfo Interface
ICatalogDomainInfo Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainManager is a factory for ICatalogDomainInfo objects used to connect to a catalog service domain. Use the CatalogDomainManager to retrieve an ICatalogDomainMananager instance.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICatalogDomainManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainManager type exposes the following members.

Methods

Name	Description
CreateCatalogDomainInfo	Creates an instance of the ICatalogDomainInfo object for the specified catalog service addresses. The catalog service addresses are expressed in a string of the form: catalogServerAddresses ::= <catalogServiceEndpoint> [, <catalogServiceEndpoint>] catalogServiceEndpoint ::= <hostName> : <listenerPort> hostName ::= The IP address or host name of a catalog service. listenerPort ::= The listener port that the catalog service is configured to use. For example: com.acme.server1:2809,com.acme.server2:2809

Back to Top

See Also

ICatalogDomainManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICatalogDomainManager type exposes the following members.

Methods

Name	Description
CreateCatalogDomainInfo	Creates an instance of the ICatalogDomainInfo object for the specified catalog service addresses. The catalog service addresses are expressed in a string of the form: catalogServerAddresses ::= <catalogServiceEndpoint> [, <catalogServiceEndpoint>] catalogServiceEndpoint ::= <hostName> : <listenerPort> hostName ::= The IP address or host name of a catalog service. listenerPort ::= The listener port that the catalog service is configured to use. For example: com.acme.server1:2809,com.acme.server2:2809

Back to Top

See Also

ICatalogDomainManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Creates an instance of the ICatalogDomainInfo object for the specified catalog service addresses.

The catalog service addresses are expressed in a string of the form:

catalogServerAddresses ::= <catalogServiceEndpoint> [,<catalogServiceEndpoint>]

catalogServiceEndpoint ::= <hostName> : <listenerPort>

hostName ::= The IP address or host name of a catalog service.

listenerPort ::= The listener port that the catalog service is configured to use.

For example:

com.acme.server1:2809,com.acme.server2:2809

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

catalogServerAddresses
Type: System String
The catalog service addresses.

Return Value

The ICatalogDomainInfo representing the catalog service domain addresses.
See Also

ICatalogDomainManager Interface
ICatalogDomainManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The handle to a connection to a catalog service domain. An IClientConnectionContext is returned from the Connect(ICatalogDomainInfo) method when connecting to a data grid.

When finished with the data grid, use the Disconnect(IClientConnectionContext) method to disconnect from the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IClientConnectionContext Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IClientConnectionContext type exposes the following members.

Properties

Name	Description
DomainName	Retrieves the name of the catalog service domain.

Back to Top

See Also

IClientConnectionContext Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IClientConnectionContext type exposes the following members.

Properties

Name	Description
DomainName	Retrieves the name of the catalog service domain.

Back to Top

See Also

IClientConnectionContext Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the name of the catalog service domain.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

IClientConnectionContext Interface
IClientConnectionContext Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IGrid Interface IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An IGrid is the client's access to a named data grid and cooresponds to a configured ObjectGrid in the catalog service domain.

An IGrid is thread safe and should be cached and reused by the application and is valid until the connection is severed.

Use an IGrid to get map instances. Maps are defined on the data grid and have unique names within each configured ObjectGrid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

The IGrid type exposes the following members.

Methods

Name	Description
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
GetGridMapNames	Returns a list of map names that are associated with this data grid.
GetGridMapPessimisticAutoTx TKey, TValue (String)	Returns a map instance for the specified map.
GetGridMapPessimisticAutoTx TKey, TValue (String, ICredentialGenerator)	Returns a map instance for the specified map using specified credential object (userid password).
GetGridMapPessimisticTx TKey, TValue (String)	Returns a grid map instance for the specified map.
GetGridMapPessimisticTx TKey, TValue (String, ICredentialGenerator)	Returns a grid map instance for the specified map using specified credential object (userid password)

Back to Top

Properties

Name	Description
IsSecurityEnabled	Retreives a value that indicates whether data grid security is enabled.
Name	Retrieves the data grid name.
TransactionTimeout	Gets or sets the default transaction timeout for the data grid.

Back to Top

See Also

IGrid Interface
IBM.WebSphere.Caching Namespace

The IGrid type exposes the following members.

Methods

Name	Description
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
GetGridMapNames	Returns a list of map names that are associated with this data grid.
GetGridMapPessimisticAutoTx TKey, TValue (String)	Returns a map instance for the specified map.
GetGridMapPessimisticAutoTx TKey, TValue (String, ICredentialGenerator)	Returns a map instance for the specified map using specified credential object (userid password).
GetGridMapPessimisticTx TKey, TValue (String)	Returns a grid map instance for the specified map.
GetGridMapPessimisticTx TKey, TValue (String, ICredentialGenerator)	Returns a grid map instance for the specified map using specified credential object (userid password)

Back to Top

See Also

IGrid Interface
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a list of map names that are associated with this data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Returns a list of map names.

See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

	Name	Description
	GetGridMapPessimisticAutoTx TKey, TValue (String)	Returns a map instance for the specified map.
	GetGridMapPessimisticAutoTx TKey, TValue (String, ICredentialGenerator)	Returns a map instance for the specified map using specified credential object (userid password).

Back to Top

See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map instance for the specified map.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

mapName
: Type: System String
Specifies the map name.

Type Parameters

TKey
: Specifies a generic type key.

TValue
: Specifies a generic type Value.

Return Value

An IGridMapPessimisticAutoTx<TKey,TValue> instance

Exceptions

Exception	Condition
IBM.WebSphere.Caching.MapLockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not supported by this map interface.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticAutoTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map instance for the specified map using specified credential object (userid password).

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

mapName
Type: System String
Specifies the map name.

credentialGenerator
Type: IBM.WebSphere.Caching.Security ICredentialGenerator
Specifies the Credential Generator object.

Type Parameters

TKey
Specifies a generic type key.
TValue
Specifies a generic type Value.

Return Value

An IGridMapPessimisticAutoTx<TKey,TValue> instance
Exceptions

Exception	Condition
IBM.WebSphere.Caching.Map LockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not supported by this map interface.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticAutoTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GetGridMapPessimisticTx TKey, TValue (String)	Returns a grid map instance for the specified map.
GetGridMapPessimisticTx TKey, TValue (String, ICredentialGenerator)	Returns a grid map instance for the specified map using specified credential object (userid password)

Back to Top

See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a grid map instance for the specified map.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

mapName
: Type: System String
: mapName is the name of the specified map

Type Parameters

TKey
: Generic type key.

TValue
: Generic Type Value.

Return Value

An IGridMapPessimisticTx<TKey,TValue> instance

Exceptions

Exception	Condition
IBM.WebSphere.Caching.MapLockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not configured for pessimistic locking.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a grid map instance for the specified map using specified credential object (userid password)

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

mapName
Type: System String
mapName is the name of the specified map

credentialGenerator
Type: IBM.WebSphere.Caching.Security ICredentialGenerator
Specifies the Credential Generator object.

Type Parameters

TKey
Generic type key.
TValue
Generic Type Value.

Return Value

An IGridMapPessimisticTx<TKey,TValue> instance
Exceptions

Exception	Condition
IBM.WebSphere.Caching.Map LockStrategyNotSupportedException	Occurs when the locking strategy for mapName is not configured for pessimistic locking.

See Also

IGrid Interface
IGrid Members
GetGridMapPessimisticTx Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGrid type exposes the following members.
Properties

	Name	Description
	IsSecurityEnabled	Retreives a value that indicates whether data grid security is enabled.
	Name	Retrieves the data grid name.
	TransactionTimeout	Gets or sets the default transaction timeout for the data grid.

Back to Top
See Also

IGrid Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retreives a value that indicates whether data grid security is enabled.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if data grid security is enabled; otherwise, false.
See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the data grid name.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The data grid name.
See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the default transaction timeout for the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The default transaction timeout value for the data grid.
See Also

IGrid Interface
IGrid Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IGridManager Interface IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Provides methods for connecting to data grids. Use the GridManagerFactory to obtain an instance to an IGridManager instance.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Examples

The following example illustrates how to retrieve an IGridManager and disconnect:
Copy to ClipboardPrint

```
// Retrieve the singleton IGridManager instance.
IGridManager gm = GridManagerFactory.GetGridManager();

// Retreive and cache the grid, and retreive and update data....
...

// Disconnect from the data grid when all done and
// null out any references.
gm.Disconnect(ccc);
grid = null;
ccc = null;
```

See Also

IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IGridManager Members IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridManager type exposes the following members.

Methods

Name	Description
Connect(ICatalogDomainInfo)	Connects a client process to the data grid. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Connect(ICatalogDomainInfo, String)	Connects a client process to the data grid using connection properties that are specified in the the configFilename. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Disconnect	Disconnects a client process from a catalog server. The IClientConnectionContext and associated IGrid instances are invalid once disconnected.
GetGrid	Returns a client data grid instance that corresponds to the data grid that is identified by the specified name. The IGrid is thread safe and should be cached for the life of the application. A grid name cooresponds to an ObjectGrid name in the data grid configuration.

Back to Top

Properties

Name	Description
CatalogDomainManager	Retrieves the CatalogDomainManager, which is used for creating ICatalogDomainInfo objects.

Back to Top

See Also

IGridManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridManager type exposes the following members.
Methods

Name	Description
Connect(ICatalogDomainInfo)	Connects a client process to the data grid. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Connect(ICatalogDomainInfo, String)	Connects a client process to the data grid using connection properties that are specified in the the configFilename. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Disconnect	Disconnects a client process from a catalog server. The IClientConnectionContext and associated IGrid instances are invalid once disconnected.
GetGrid	Returns a client data grid instance that corresponds to the data grid that is identified by the specified name. The IGrid is thread safe and should be cached for the life of the application. A grid name cooresponds to an ObjectGrid name in the data grid configuration.

Back to Top
See Also

IGridManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Connect(ICatalogDomainInfo)	Connects a client process to the data grid. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.
Connect(ICatalogDomainInfo, String)	Connects a client process to the data grid using connection properties that are specified in the the configFilename. Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.

Back to Top

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Connects a client process to the data grid.

Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

catalogDomainInfo
Type: IBM.WebSphere.Caching ICatalogDomainInfo
Specifies the CatalogDomainInfo object that contains target catalog service domain information.

Return Value

Returns an IClientConnectionContext object that represents a handle to the data grid to which the client is connected.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullcatalogDomainInfo is specified.
IBM.WebSphere.Caching GridException	Occurs if there is an error connecting to the specified catalog domain.

See Also

IGridManager Interface
IGridManager Members
Connect Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Connects a client process to the data grid using connection properties that are specified in the the configFilename.

Multiple invocations will produce additional connections. The IClientConnectionContext should be cached for the life of the application, and the Disconnect(IClientConnectionContext) method should be invoked to disconnect from the data grid.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

catalogDomainInfo
Type: IBM.WebSphere.Caching ICatalogDomainInfo
Specifies the CatalogDomainInfo object that contains target catalog service domain information.

configFilename
Type: System String
Specifies the full path and file name of the client configuration file or null to use the default configuration.

Return Value

Returns an IClientConnectionContext object that represents a handle to the data grid to which the client is connected.
Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullcatalogDomainInfo is specified.
IBM.WebSphere.Caching GridException	Occurs if there is an error connecting to the specified catalog domain.
System.IO FileNotFoundException	Occurs if configFilename does not exist.

See Also

IGridManager Interface
IGridManager Members
Connect Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Disconnects a client process from a catalog server. The IClientConnectionContext and associated IGrid instances are invalid once disconnected.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

context
: Type: IBM.WebSphere.Caching IClientConnectionContext
Specifies the IClientConnectionContext object that was returned from a previous Connect method call.

Return Value

Returns true if the disconnect was successful, or false if the supplied context was not connected.

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a client data grid instance that corresponds to the data grid that is identified by the specified name. The IGrid is thread safe and should be cached for the life of the application.

A grid name cooresponds to an ObjectGrid name in the data grid configuration.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

context
: Type: IBM.WebSphere.Caching IClientConnectionContext
: Specifies the IClientConnectionContext object returned from a Connect method call.

gridName
: Type: System String
: Specifies the name of the requested ObjectGrid.

Return Value

Returns client data grid instance that corresponds to the specified remote data grid.
Exceptions

Exception	Condition
IBM.WebSphere.Caching GridException	Occurs if the specified grid is not found or incompatible with the current configuration.

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridManager type exposes the following members.
Properties

	Name	Description
	CatalogDomainManager	Retrieves the CatalogDomainManager, which is used for creating ICatalogDomainInfo objects.

Back to Top
See Also

IGridManager Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the CatalogDomainManager, which is used for creating ICatalogDomainInfo objects.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGridManager Interface
IGridManager Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Defines the interface for a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridTransaction type exposes the following members.

Methods

Name	Description
Begin	Begins a new transaction.
Commit	Commits a transaction.
Flush	Apply the current changes to the server without committing the transaction.
MarkRollbackOnly	Marks the current transaction as being rollback only.
Rollback	Rolls back a transaction.

Back to Top

Properties

Name	Description
Active	Determines if a transaction is currently active.
MarkedRollbackOnly	Determines if a transaction is roll back only.
TransactionIsolationLevel	Gets or sets the transaction isolation level.
TransactionTimeout	Gets or sets the amount of time in which a transaction must complete. To use an unlimited transaction timeout value, set the value to TimeSpan.Zero

Back to Top

See Also

IGridTransaction Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridTransaction type exposes the following members.

Methods

Name	Description
Begin	Begins a new transaction.
Commit	Commits a transaction.
Flush	Apply the current changes to the server without committing the transaction.
MarkRollbackOnly	Marks the current transaction as being rollback only.
Rollback	Rolls back a transaction.

Back to Top

See Also

IGridTransaction Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Begins a new transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Commits a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Apply the current changes to the server without committing the transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Marks the current transaction as being rollback only.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

error
Type: System Exception
Specifies the cause of the exception.

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Rolls back a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridTransaction type exposes the following members.

Properties

Name	Description
Active	Determines if a transaction is currently active.
MarkedRollbackOnly	Determines if a transaction is roll back only.
TransactionIsolationLevel	Gets or sets the transaction isolation level.
TransactionTimeout	Gets or sets the amount of time in which a transaction must complete. To use an unlimited transaction timeout value, set the value to TimeSpan.Zero

Back to Top

See Also

IGridTransaction Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines if a transaction is currently active.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines if a transaction is roll back only.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the transaction isolation level.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The transaction isolation level.

See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the amount of time in which a transaction must complete. To use an unlimited transaction timeout value, set the value to TimeSpan.Zero

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The transaction timeout value.
See Also

IGridTransaction Interface
IGridTransaction Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies a generic version of the non-generic IOrderedDictionary interface.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Type Parameters

TKey
: Specifies the key type.

TValue
: Specifies the value type.

See Also

IOrderedDictionary TKey, TValue _Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IOrderedDictionary TKey, TValue type exposes the following members.

Methods

Name	Description
Add(T)	Adds an item to the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Add(TKey, TValue)	Adds an element with the provided key and value to the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Clear	Removes all items from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Contains	Determines whether the ICollection T contains a specific value. (Inherited from ICollection KeyValuePair TKey, TValue .)
ContainsKey	Determines whether the IDictionary TKey, TValue contains an element with the specified key. (Inherited from IDictionary TKey, TValue .)
CopyTo	Copies the elements of the ICollection T to an Array, starting at a particular Array index. (Inherited from ICollection KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through the collection. (Inherited from IEnumerable KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through a collection. (Inherited from IEnumerable.)
Insert	Inserts an item with the specified key and value into the map at the specified index.
Remove(TKey)	Removes the element with the specified key from the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Remove(T)	Removes the first occurrence of a specific object from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
RemoveAt	Removes the item at the specified index from the IOrderedDictionary object.
TryGetValue	Gets the value associated with the specified key. (Inherited from IDictionary TKey, TValue .)

Back to Top

Properties

Name	Description
Count	Gets the number of elements contained in the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
IsReadOnly	Gets a value indicating whether the ICollection T is read-only. (Inherited from ICollection KeyValuePair TKey, TValue .)
Item TKey	Gets or sets the element with the specified key. (Inherited from IDictionary TKey, TValue .)
Item Int32	Gets or sets the value with the specified index.
Keys	Gets an ICollection T containing the keys of the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)

Values	Gets an ICollection T containing the values in the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)

Back to Top
See Also

IOrderedDictionary TKey, TValue Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IOrderedDictionary TKey, TValue type exposes the following members.

Methods

Name	Description
Add(T)	Adds an item to the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Add(TKey, TValue)	Adds an element with the provided key and value to the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Clear	Removes all items from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
Contains	Determines whether the ICollection T contains a specific value. (Inherited from ICollection KeyValuePair TKey, TValue .)
ContainsKey	Determines whether the IDictionary TKey, TValue contains an element with the specified key. (Inherited from IDictionary TKey, TValue .)
CopyTo	Copies the elements of the ICollection T to an Array, starting at a particular Array index. (Inherited from ICollection KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through the collection. (Inherited from IEnumerable KeyValuePair TKey, TValue .)
GetEnumerator	Returns an enumerator that iterates through a collection. (Inherited from IEnumerable.)
Insert	Inserts an item with the specified key and value into the map at the specified index.
Remove(TKey)	Removes the element with the specified key from the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
Remove(T)	Removes the first occurrence of a specific object from the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
RemoveAt	Removes the item at the specified index from the IOrderedDictionary object.
TryGetValue	Gets the value associated with the specified key. (Inherited from IDictionary TKey, TValue .)

Back to Top

See Also

IOrderedDictionary TKey, TValue Interface
IBM.WebSphere.Caching Namespace

Inserts an item with the specified key and value into the map at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
Specifies the zero-based index where the item should be inserted

key
Type: TKey
Specifies the key of the item to insert.

value
Type: TValue
Specifies the value of the item to insert.

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Occurs when index is out of range.
System ArgumentNullException	Occurs when a null key is specified.
System ArgumentException	Occurs when key already exists in the IOrderedDictionary object.

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the item at the specified index from the IOrderedDictionary object.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index of the item to remove.

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Occurs when index is less than 0 or greater than Count value.

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IOrderedDictionary TKey, TValue type exposes the following members.

Properties

	Name	Description
	Count	Gets the number of elements contained in the ICollection T . (Inherited from ICollection KeyValuePair TKey, TValue .)
	IsReadOnly	Gets a value indicating whether the ICollection T is read-only. (Inherited from ICollection KeyValuePair TKey, TValue .)
	Item TKey	Gets or sets the element with the specified key. (Inherited from IDictionary TKey, TValue .)
	Item Int32	Gets or sets the value with the specified index.
	Keys	Gets an ICollection T containing the keys of the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)
	Values	Gets an ICollection T containing the values in the IDictionary TKey, TValue . (Inherited from IDictionary TKey, TValue .)

Back to Top

See Also

IOrderedDictionary TKey, TValue Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

	Name	Description
	Item TKey	Gets or sets the element with the specified key. (Inherited from IDictionary TKey, TValue .)
	Item Int32	Gets or sets the value with the specified index.

Back to Top

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Gets or sets the value with the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

index
Type: System Int32
Specifies the zero-based index of the value to get or set.

Return Value

The value of the item at the specified index.

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Occurs when index is less than 0 or greater than the Count value.

See Also

IOrderedDictionary TKey, TValue Interface
IOrderedDictionary TKey, TValue Members
Item Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Implementors of ITransactionable provide transaction demarcation semantics.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ITransactionable Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ITransactionable type exposes the following members.

Properties

Name	Description
Transaction	The Transaction instance used to configure and demarcate a transaction.

Back to Top

See Also

ITransactionable Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ITransactionable type exposes the following members.

Properties

Name	Description
Transaction	The Transaction instance used to configure and demarcate a transaction.

Back to Top

See Also

ITransactionable Interface
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The Transaction instance used to configure and demarcate a transaction.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ITransactionable Interface
ITransactionable Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LifecycleFailedException exception occurs on unexpected lifecycle states.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching LifecycleFailedException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LifecycleFailedException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LifecycleFailedException type exposes the following members.

Constructors

Name	Description
LifecycleFailedException(String)	Initializes a new instance of the LifecycleFailedException class with the specified error message.
LifecycleFailedException(String, Exception)	Initializes a new instance of the LifecycleFailedException class with the specified error message and exception cause.

Back to Top

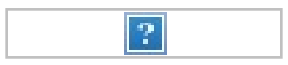
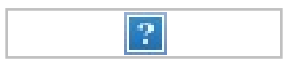
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)

	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LifecycleFailedException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LifecycleFailedException(String)	Initializes a new instance of the LifecycleFailedException class with the specified error message.
LifecycleFailedException(String, Exception)	Initializes a new instance of the LifecycleFailedException class with the specified error message and exception cause.

Back to Top

See Also

LifecycleFailedException Class
LifecycleFailedException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LifecycleFailedException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LifecycleFailedException Class
LifecycleFailedException Members
LifecycleFailedException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LifecycleFailedException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the exception message text.

cause
Type: System Exception
Specifies the initial cause of the exception that caused this exception to occur.

See Also

LifecycleFailedException Class
LifecycleFailedException Members
LifecycleFailedException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LifecycleFailedException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LifecycleFailedException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LifecycleFailedException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LifecycleFailedException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A MixedTransportException is thrown when server and client have mismatched transport. For example, a server is using ORB, but the client is eXtremeIO

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching MixedTransportException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

MixedTransportException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MixedTransportException type exposes the following members.

Constructors

Name	Description
MixedTransportException	Initializes a new instance of the MixedTransportException class.
MixedTransportException(String)	Initializes a new instance of the MixedTransportException class with the specified error message.

Back to Top

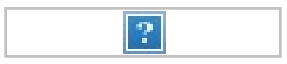
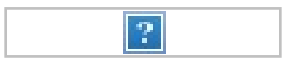
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)

	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

MixedTransportException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
MixedTransportException	Initializes a new instance of the MixedTransportException class.
MixedTransportException(String)	Initializes a new instance of the MixedTransportException class with the specified error message.

Back to Top

See Also

MixedTransportException Class
MixedTransportException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MixedTransportException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

MixedTransportException Class
MixedTransportException Members
MixedTransportException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MixedTransportException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

MixedTransportException Class
MixedTransportException Members
MixedTransportException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MixedTransportException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

MixedTransportException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MixedTransportException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

MixedTransportException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A NoActiveTransactionException exception indicates that no active transactions exist.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionException
        IBM.WebSphere.Caching NoActiveTransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

NoActiveTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Constructors

Name	Description
NoActiveTransactionException	Initializes a new instance of the NoActiveTransactionException class.
NoActiveTransactionException(Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified exception cause.
NoActiveTransactionException(String)	Initializes a new instance of the NoActiveTransactionException class with the specified error message.
NoActiveTransactionException(String, Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

Fields

Name	Description
ivTransactionRolledBa	Indicates whether the transaction was rolled back or

ck	not. (Inherited from TransactionException.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
NoActiveTransactionException	Initializes a new instance of the NoActiveTransactionException class.
NoActiveTransactionException(Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified exception cause.
NoActiveTransactionException(String)	Initializes a new instance of the NoActiveTransactionException class with the specified error message.
NoActiveTransactionException(String, Exception)	Initializes a new instance of the NoActiveTransactionException class with the specified error message and exception cause.

Back to Top

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NoActiveTransactionException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NoActiveTransactionException Class
NoActiveTransactionException Members
NoActiveTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not. (Inherited from TransactionException.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NoActiveTransactionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NoActiveTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A NotReentrantException occurs when a thread tries to run a map operation, such as calling a method on ObjectMap interface, when another thread is already running a map operation for the Session. A Session object can be used by a single thread only to perform concurrent map operations.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching NotReentrantException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

NotReentrantException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NotReentrantException type exposes the following members.

Constructors

Name	Description
NotReentrantException	Initializes a new instance of the NotReentrantException class.
NotReentrantException(Exception)	Initializes a new instance of the NotReentrantException class with a specified exception cause.
NotReentrantException(String)	Initializes a new instance of the NotReentrantException class with the specified error message.
NotReentrantException(String, Exception)	Initializes a new instance of the NotReentrantException class with the specified error message and exception cause.

Back to Top

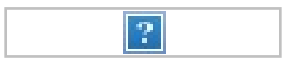
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception.

		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NotReentrantException Class

IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
NotReentrantException	Initializes a new instance of the NotReentrantException class.
NotReentrantException(Exception)	Initializes a new instance of the NotReentrantException class with a specified exception cause.
NotReentrantException(String)	Initializes a new instance of the NotReentrantException class with the specified error message.
NotReentrantException(String, Exception)	Initializes a new instance of the NotReentrantException class with the specified error message and exception cause.

Back to Top

See Also

NotReentrantException Class
NotReentrantException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the NotReentrantException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

NotReentrantException Class
NotReentrantException Members
NotReentrantException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NotReentrantException type exposes the following members.

Methods

	Name	Description
	Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
	Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
	GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
	GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

NotReentrantException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The NotReentrantException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

NotReentrantException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A generic version of the non-generic OrderedDictionary class.

Inheritance Hierarchy

System Object
  IBM.WebSphere.Caching OrderedDictionary TKey, TValue

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
    The key type
TValue
    The value type

See Also

OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue type exposes the following members.

Constructors

Name	Description
OrderedDictionary TKey, TValue	Initializes a new instance of the OrderedDictionary TKey, TValue class.
OrderedDictionary TKey, TValue (Int32)	Initializes a new instance of the OrderedDictionary TKey, TValue class with the specified capacity.
OrderedDictionary TKey, TValue (IOrderedDictionary TKey, TValue)	Initializes a new instance of the OrderedDictionary TKey, TValue class having its contents copied from the specified IOrderedDictionary TKey, TValue .

Back to Top

Methods

Name	Description
Add(KeyValuePair TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Add(TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Clear	Removes all items from the OrderedDictionary TKey, TValue .
Contains	Determines whether the OrderedDictionary TKey, TValue contains the specified item.
ContainsKey	Determines whether the OrderedDictionary TKey, TValue contains the specified key.
CopyTo(Array, Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.
CopyTo(KeyValue Pair TKey, TValue , Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetEnumerator	Returns a Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
Insert(Int32, KeyValuePair TKey , TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
Insert(Int32, TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)

Remove(KeyValuePair TKey, TValue)	Removes the specified item from the OrderedDictionary TKey, TValue .
Remove(TKey)	Removes the item with the specified key from the OrderedDictionary TKey, TValue .
RemoveAt	Removes the item at the specified index from the OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)
TryGetValue	Get the value associated with the specified key.

Back to Top

Properties

Name	Description
Count	Gets the number of items in the OrderedDictionary TKey, TValue .
IsFixedSize	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue has a fixed size.
IsReadOnly	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue is read only.
IsSynchronized	Gets a value indicating whether access to this Collection is synchronized (thread safe).
Item Int32	Gets or sets the value associated with the specified index.
Item TKey	Gets or sets the value associated with the specified key.
Keys	Gets a ICollection T object containing all the keys in the OrderedDictionary TKey, TValue .
SyncRoot	Gets an object that can be used to synchronize this Collection.
Values	Gets a ICollection T object containing all the values in the OrderedDictionary TKey, TValue .

Back to Top

See Also

OrderedDictionary TKey, TValue Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
OrderedDictionary TKey, TValue	Initializes a new instance of the OrderedDictionary TKey, TValue class.
OrderedDictionary TKey, TValue (Int32)	Initializes a new instance of the OrderedDictionary TKey, TValue class with the specified capacity.
OrderedDictionary TKey, TValue (IOrderedDictionary TKey, TValue)	Initializes a new instance of the OrderedDictionary TKey, TValue class having its contents copied from the specified IOrderedDictionary TKey, TValue.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OrderedDictionary TKey, TValue class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
OrderedDictionary TKey, TValue Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OrderedDictionary TKey, TValue class with the specified capacity.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

capacity
Type: System Int32
The initial number of elements that OrderedDictionary TKey, TValue can contain.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
OrderedDictionary TKey, TValue Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OrderedDictionary TKey, TValue class having its contents copied from the specified IOrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

orderedDictionary
Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
The IOrderedDictionary TKey, TValue whose contents are copied into the new OrderedDictionary TKey, TValue .

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
OrderedDictionary TKey, TValue Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue type exposes the following members.

Methods

Name	Description
Add(KeyValuePair TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Add(TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Clear	Removes all items from the OrderedDictionary TKey, TValue .
Contains	Determines whether the OrderedDictionary TKey, TValue contains the specified item.
ContainsKey	Determines whether the OrderedDictionary TKey, TValue contains the specified key.
CopyTo(Array, Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.
CopyTo(KeyValue Pair TKey, TValue , Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetEnumerator	Returns a Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
Insert(Int32, KeyValuePair TKey , TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
Insert(Int32, TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
Remove(KeyValue Pair TKey, TValue)	Removes the specified item from the OrderedDictionary TKey, TValue .
Remove(TKey)	Removes the item with the specified key from the OrderedDictionary TKey, TValue .
RemoveAt	Removes the item at the specified index from the OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)
TryGetValue	Get the value associated with the specified key.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Add(KeyValuePair TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .
Add(TKey, TValue)	Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The key-value pair to add.

Implements

ICollection T Add(T)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.
System.Collections.Generic KeyNotFoundException	Thrown when the kvp key is not found.

Remarks

Use this property to add value in the collection. Values are appended to the end of the collection. A key cannot be null; however a value can be. To update an existing item, use either the Item Int32 or the Item TKey property.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Add Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds an item with the specified key and value to the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the item to add.

value
Type: TValue
The value of the item to add.

Implements

IDictionary TKey, TValue Add(TKey, TValue)
Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

Remarks

Use this property to add a new item to the collection. Items are appended to the end of the collection. A key cannot be null; however a value can be. To update an existing item, use either the Item Int32 or the Item Int32 property.
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Add Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes all items from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Implements

ICollection T Clear
IDictionary Clear
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines whether the OrderedDictionary TKey, TValue contains the specified item.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The key-value pair to locate in the OrderedDictionary TKey, TValue .

Return Value

true if the OrderedDictionary TKey, TValue contains the item associated with the specified kvp; otherwise, false.

Implements

ICollection T Contains(T)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.

Remarks

The specified item's key must be contained in the OrderedDictionary TKey, TValue and it's associated value must be equal to the specified item's value for true to be returned.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Determines whether the OrderedDictionary TKey, TValue contains the specified key.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

key
Type: TKey
The key to locate in the OrderedDictionary TKey, TValue .

Return Value

true if the OrderedDictionary TKey, TValue contains the item associated with the specified key; otherwise, false.

Implements

IDictionary TKey, TValue ContainsKey(TKey)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
CopyTo(Array, Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.
CopyTo(KeyValuePair TKey, TValue , Int32)	Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Copies the items in the OrderedDictionary TKey, TValue into the specified single-dimension array at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

array
Type: System Array
The allocated single-dimension array into which the items from OrderedDictionary TKey, TValue will be copied.

index
Type: System Int32
The index into array where copying will begin.

Implements

ICollection CopyTo(Array, Int32)

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is out of range.
System ArgumentException	Thrown when array does not have sufficient size to hold the incoming data.

Remarks

The array must have sufficient storage to contain all items from the OrderedDictionary TKey, TValue ; otherwise a ArgumentException will be thrown. index must be equal or greater than 0.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
CopyTo Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Copies the items in the OrderedDictionary TKey, TValue into the specified array at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

array
Type: System.Collections.Generic KeyValuePair TKey, TValue
The allocated array into which the items from OrderedDictionary TKey, TValue will be copied.

index
Type: System Int32
The zero-based index into array where copying will begin.

Implements

ICollection T CopyTo(T , Int32)

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is out of range.
System ArgumentException	Thrown when array does not have sufficient size to hold the incoming data.

Remarks

The array must have sufficient storage to contain all items from the OrderedDictionary TKey, TValue ; othewise a ArgumentException will be thrown. index must be equal or greater than 0.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
CopyTo Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

A Systems.Collections.Generic.IEnumerator<KeyValuePair<TKey,TValue>> object that iterates through the OrderedDictionary TKey, TValue .

Implements

IEnumerable T GetEnumerator
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Insert(Int32, KeyValuePair TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.
Insert(Int32, TKey, TValue)	Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index where the item should be inserted.

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The item to insert.

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.
System ArgumentOutOfRangeException	Thrown when index is out of range.

Remarks

The index must be greater than or equal to 0 and the index must be less than the number of elements in the collection; otherwise a ArgumentOutOfRangeException is thrown.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Insert Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Inserts an item with the specified key and value into the OrderedDictionary TKey, TValue at the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index where the item should be inserted.

key
Type: TKey
The key of the item to insert.

value
Type: TValue
The value of the item to insert.

Implements

IOrderedDictionary TKey, TValue Insert(Int32, TKey, TValue)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the key is null.
System ArgumentOutOfRangeException	Thrown when index is out of range.

Remarks

The index must be greater than or equal to 0 and the index must be less than the number of elements in the collection; otherwise a ArgumentOutOfRangeException is thrown.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Insert Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Remove(KeyValuePair TKey, TValue)	Removes the specified item from the OrderedDictionary TKey, TValue .
Remove(TKey)	Removes the item with the specified key from the OrderedDictionary TKey, TValue .

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the specified item from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

kvp
Type: System.Collections.Generic KeyValuePair TKey, TValue
The key-value pair to remove.

Return Value

true if the item is successfully removed; otherwise false is returned. false is also returned if the item is not found.

Implements

ICollection T Remove(T)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when the kvp key is null.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Remove Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the item with the specified key from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the item to remove.

Return Value

true if the item is successfully removed; otherwise false. false is also returned if the key is not found.

Implements

IDictionary TKey, TValue Remove(TKey)
Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Remove Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the item at the specified index from the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index of the item to remove.

Implements

IOrderedDictionary TKey, TValue _RemoveAt(Int32)

Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is out of range.

Remarks

The index must be greater than or equal to 0 and the index must be less than the number of elements in the collection; otherwise a ArgumentOutOfRangeException is thrown.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Get the value associated with the specified key.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Parameters

key
Type: TKey
The key of the value to get.

val
Type: TValue
The value associated with the specified key.

Return Value

true if the OrderedDictionary TKey, TValue contained the specified key; otherwise false.

Implements

IDictionary TKey, TValue TryGetValue(TKey, TValue)

Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.

Remarks

If the key is not found, val is initialized to the default value associated with its data type.

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue type exposes the following members.

Properties

	Name	Description
	Count	Gets the number of items in the OrderedDictionary TKey, TValue.
	IsFixedSize	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue has a fixed size.
	IsReadOnly	Gets a value that indicates whether or not the OrderedDictionary TKey, TValue is read only.
	IsSynchronized	Gets a value indicating whether access to this Collection is synchronized (thread safe).
	Item Int32	Gets or sets the value associated with the specified index.
	Item TKey	Gets or sets the value associated with the specified key.
	Keys	Gets a ICollection T object containing all the keys in the OrderedDictionary TKey, TValue.
	SyncRoot	Gets an object that can be used to synchronize this Collection.
	Values	Gets a ICollection T object containing all the values in the OrderedDictionary TKey, TValue.

Back to Top

See Also

OrderedDictionary TKey, TValue Class

IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the number of items in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The number of items in the OrderedDictionary TKey, TValue .

Implements

ICollection T Count
ICollection Count
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Gets a value that indicates whether or not the OrderedDictionary TKey, TValue has a fixed size.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if the OrderedDictionary TKey, TValue has a fixed size; otherwise false.

Implements

IDictionary IsFixedSize
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets a value that indicates whether or not the OrderedDictionary TKey, TValue is read only.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if the OrderedDictionary TKey, TValue is read only; otherwise false.

Implements

ICollection T IsReadOnly
IDictionary IsReadOnly
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets a value indicating whether access to this Collection is synchronized (thread safe).

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

true if this Collection is synchronized.

Implements

ICollection IsSynchronized

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
Item Int32	Gets or sets the value associated with the specified index.
Item TKey	Gets or sets the value associated with the specified key.

Back to Top

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Gets or sets the value associated with the specified index.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

index
Type: System Int32
The zero-based index of the value to get or set.

Field Value

The value of the item at the specified index.

Implements

IOrderedDictionary TKey, TValue Item Int32
Exceptions

Exception	Condition
System ArgumentOutOfRangeException	Thrown when index is less than 0 or greater than the number of elements in the collection.

Remarks

A set operation will add or replace a value in the collection, with new values being appended to the end of the collection. A key must be greater than or equal to 0. Values may be null.
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Item Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the value associated with the specified key.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the value to get or set.

Field Value

The value associated with the specified key. If the specified key does not exist during a get operation, a KeyNotFoundException is thrown.

Implements

IDictionary TKey, TValue Item TKey
Exceptions

Exception	Condition
System ArgumentNullException	Thrown when key is null.
System.Collections.Generic KeyNotFoundException	Thrown when key is not found during a get operation.

Remarks

A set operation will add or replace a value in the collection, with new values being appended to the end of the collection. A key cannot be null; however a value can be. The key cannot be an int; if it is an int, then the Item Int32 property is used.
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
Item Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets a ICollection T object containing all the keys in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

A ICollection T object containing all the keys in the OrderedDictionary TKey, TValue .

Implements

IDictionary TKey, TValue Keys

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets an object that can be used to synchronize this Collection.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Field Value

An object that can be used to synchronize this Collection.

Implements

ICollection SyncRoot

See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

Gets a ICollection T object containing all the values in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

A ICollection T object containing all the values in the OrderedDictionary TKey, TValue .

Implements

IDictionary TKey, TValue Values
See Also

OrderedDictionary TKey, TValue Class
OrderedDictionary TKey, TValue Members
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The enumerator for interating through the OrderedDictionary TKey, TValue .

Inheritance Hierarchy

System Object
  IBM.WebSphere.Caching OrderedDictionary TKey, TValue Enumerator

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A generic version of the non-generic OrderedDictionary class.

The OrderedDictionary TKey, TValue Enumerator generic type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
MoveNext	Moves the iterator to the next item in the OrderedDictionary TKey, TValue .
Reset	Moves the iterator to the first item in OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
Current	Gets the item at the iterator's current position.

Back to Top

See Also

OrderedDictionary TKey, TValue Enumerator Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue Enumerator generic type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
MoveNext	Moves the iterator to the next item in the OrderedDictionary TKey, TValue .
Reset	Moves the iterator to the first item in OrderedDictionary TKey, TValue .
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

OrderedDictionary TKey, TValue Enumerator Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Moves the iterator to the next item in the OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

true if the enumerator was successfully able to move the next item in the OrderedDictionary TKey, TValue ; otherwise, false is returned to indicate that the enumerator is at the end of the OrderedDictionary TKey, TValue .

Implements

IEnumerator MoveNext

See Also

OrderedDictionary TKey, TValue Enumerator Class
OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Moves the iterator to the first item in OrderedDictionary TKey, TValue .

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Implements

IEnumerator Reset

See Also

OrderedDictionary TKey, TValue Enumerator Class
OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OrderedDictionary TKey, TValue Enumerator generic type exposes the following members.

Properties

Name	Description
Current	Gets the item at the iterator's current position.

Back to Top

See Also

OrderedDictionary TKey, TValue Enumerator Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the item at the iterator's current position.

Namespace: IBM.WebSphere.Caching
Assembly: Client.ApiImpl (in Client.ApiImpl.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Implements

IEnumerator T Current

See Also

OrderedDictionary TKey, TValue Enumerator Class
OrderedDictionary TKey, TValue Enumerator Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ReplicationVotedToRollbackTransactionException exception occurs when a transaction was rolled back because some or all of the synchronous replicas did not apply the transaction.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionCallbackException
        IBM.WebSphere.Caching ReplicationVotedToRollbackTransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ReplicationVotedToRollbackTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReplicationVotedToRollbackTransactionException type exposes the following members.

Constructors

Name	Description
ReplicationVotedToRollbackTransactionException	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class.
ReplicationVotedToRollbackTransactionException(Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with a specified exception cause.
ReplicationVotedToRollbackTransactionException(String)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message.
ReplicationVotedToRollbackTransactionException(String, Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ReplicationVotedToRollbackTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ReplicationVotedToRollbackTransactionException	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class.
ReplicationVotedToRollbackTransactionException(Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with a specified exception cause.
ReplicationVotedToRollbackTransactionException(String)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message.
ReplicationVotedToRollbackTransactionException(String, Exception)	Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message and exception cause.

Back to Top

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
The exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReplicationVotedToRollbackTransactionException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ReplicationVotedToRollbackTransactionException Class
ReplicationVotedToRollbackTransactionException Members
ReplicationVotedToRollbackTransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReplicationVotedToRollbackTransactionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ReplicationVotedToRollbackTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReplicationVotedToRollbackTransactionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ReplicationVotedToRollbackTransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionAlreadyActiveException exception occurs to indicate that a transaction is already active for the current Session. This exception does not cause the current active transaction to be rolled back, so the isTransactionActive method returns true.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionException
        IBM.WebSphere.Caching TransactionAlreadyActiveException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionAlreadyActiveException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Constructors

Name	Description
TransactionAlreadyActiveException	Initializes a new instance of the TransactionAlreadyActiveException class.
TransactionAlreadyActiveException(Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified exception cause.
TransactionAlreadyActiveException(String)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message.
TransactionAlreadyActiveException(String, Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

Fields

Name	Description
ivTransactionRolledBa	Indicates whether the transaction was rolled back or

ck	not. (Inherited from TransactionException.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionAlreadyActiveException	Initializes a new instance of the TransactionAlreadyActiveException class.
TransactionAlreadyActiveException(Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified exception cause.
TransactionAlreadyActiveException(String)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message.
TransactionAlreadyActiveException(String, Exception)	Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message and exception cause.

Back to Top

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAlreadyActiveException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionAlreadyActiveException Class
TransactionAlreadyActiveException Members
TransactionAlreadyActiveException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not. (Inherited from TransactionException.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed. (Inherited from TransactionException.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back. (Inherited from TransactionException.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAlreadyActiveException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionAlreadyActiveException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionCallbackException exception occurs when a TransactionCallback method call fails.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionCallbackException
        IBM.WebSphere.Caching ClientServerTransactionCallbackException
        IBM.WebSphere.Caching ReplicationVotedToRollbackTransactionException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionCallbackException type exposes the following members.

Constructors

Name	Description
TransactionCallbackException	Initializes a new instance of the TransactionCallbackException class.
TransactionCallbackException(Exception)	Initializes a new instance of the TransactionCallbackException class with a specified exception cause.
TransactionCallbackException(String)	Initializes a new instance of the TransactionCallbackException class with the specified error message.
TransactionCallbackException(String, Exception)	Initializes a new instance of the TransactionCallbackException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

TransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionCallbackException	Initializes a new instance of the TransactionCallbackException class.
TransactionCallbackException(Exception)	Initializes a new instance of the TransactionCallbackException class with a specified exception cause.
TransactionCallbackException(String)	Initializes a new instance of the TransactionCallbackException class with the specified error message.
TransactionCallbackException(String, Exception)	Initializes a new instance of the TransactionCallbackException class with the specified error message and exception cause.

Back to Top

See Also

TransactionCallbackException Class
TransactionCallbackException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class with the specified error message.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionCallbackException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionCallbackException Class
TransactionCallbackException Members
TransactionCallbackException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionCallbackException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionCallbackException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionCallbackException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionException exception is a general locking exception that indicates something went wrong with a transaction. Use the isTransactionActive() and wasTransactionRolledBack() methods to determine whether transaction is still active or was rolled back as a result of this exception.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching TransactionException
        IBM.WebSphere.Caching NoActiveTransactionException
        IBM.WebSphere.Caching TransactionAlreadyActiveException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Constructors

Name	Description
TransactionException(Exception, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Boolean)	Initializes a new instance of the TransactionException class with the specified error message and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(TransactionException, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, TransactionException, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Exception, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back.

Back to Top

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not.

Back to Top

Properties

	Name	Description
	Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionException(Exception, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Boolean)	Initializes a new instance of the TransactionException class with the specified error message and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(TransactionException, Boolean)	Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, TransactionException, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.
TransactionException(String, Exception, Boolean)	Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Back to Top

See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with the specified error message and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

rolledBack
: Type: System Boolean
Specifies if the trasnsaction was rolled back. A value of true indicates that the transaction was rolled back.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

TransactionException Constructor (TransactionException, Boolean)

IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with a specified exception cause and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: IBM.WebSphere.Caching TransactionException
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
: Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

Remarks

The cause and a detailed message of (cause==null ? null : cause.toString()) is used, which typically contains the class and detailed message of cause. This constructor is useful for as a wrapper for other Throwable objects that occur.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: IBM.WebSphere.Caching TransactionException
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
: Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

Remarks

The detailed error message that is associated with the cause is not automatically incorporated in this the detailed message for this TransactionException exception.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionException class with the specified error message, the cause, and a special indication of whether the transaction was rolled back as a result of this exception.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

rolledBack
Type: System Boolean
Specifies if the transaction was rolled back. A value of true indicates that the transaction was rolled back.

See Also

TransactionException Class
TransactionException Members
TransactionException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Fields

	Name	Description
	ivTransactionRolledBack	Indicates whether the transaction was rolled back or not.

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Indicates whether the transaction was rolled back or not.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
isTransactionActive	Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
wasTransactionRolledBack	Returns true if the transaction was rolled back.

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns true if the transaction is active. A value of false indicates that the transaction never started or was completed.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

true if transaction is active, false if transaction never started or was complete.
See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns true if the transaction was rolled back.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Returns true if transaction was rolled back, false otherwise.

See Also

TransactionException Class
TransactionException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionTimeoutException exception occurs when a transaction exceeds the transaction timeout value that was specified on the ObjectGrid or Session.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching TransactionTimeoutException

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionTimeoutException type exposes the following members.

Constructors

Name	Description
TransactionTimeoutException(String, Exception)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and exception cause.
TransactionTimeoutException(String, String)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and the transaction that timed out.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
getTxIDString	Gets the String representation of the TxID.toString() method for the transaction that timed out.
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
whenOccurred	Returns the time when this TransactionTimeoutException exception was created.

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerExc	Gets the Exception instance that caused the current exception.

	eption	(Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

TransactionTimeoutException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionTimeoutException(String, Exception)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and exception cause.
TransactionTimeoutException(String, String)	Initializes a new instance of the TransactionTimeoutException class with the specified error message and the transaction that timed out.

Back to Top

See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionTimeoutException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
TransactionTimeoutException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionTimeoutException class with the specified error message and the transaction that timed out.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

txIdString
Type: System String
Specifies the result of the TxID.toString() method for the transaction that timed out.

See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
TransactionTimeoutException Overload
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionTimeoutException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
getTxIDString	Gets the String representation of the TxID.toString() method for the transaction that timed out.
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)
whenOccurred	Returns the time when this TransactionTimeoutException exception was created.

Back to Top

See Also

TransactionTimeoutException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the String representation of the TxID.toString() method for the transaction that timed out.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the result of TxID.toString() method for the transaction that timed out.
See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns the time when this TransactionTimeoutException exception was created.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies a date object that represents the exact time when this exception object was created.
See Also

TransactionTimeoutException Class
TransactionTimeoutException Members
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionTimeoutException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionTimeoutException Class
IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies an enumeration that defines the valid transaction isolation level values.

Namespace: IBM.WebSphere.Caching
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Members

Member name	Value	Description
NotSet	-1	Specifies that the transaction isolation level value is not set.
ReadUncommitted	1	Specifies that dirty reads, non-repeatable, reads and phantom reads can occur.
ReadCommitted	2	Specifies that dirty reads are prevented, but non-repeatable reads and phantom reads can occur.
RepeatableRead	4	Specifies that dirty reads and non-repeatable reads are prevented, but phantom reads can occur.

See Also

IBM.WebSphere.Caching Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IBM.WebSphere.Caching.Map namespace includes the data access application programming interfaces. See the IBM.WebSphere.Caching namespace documentation for a description on how to access a map.

The eXtreme Scale client supports transactional data access to individual maps using automatic and manual transactions. The following maps are available:

- The IGridMapPessimisticAutoTx TKey, TValue interface provides automatic transactions.
- The IGridMapPessimisticTx TKey, TValue interface provides manual transaction demarcation.

See each respective interface for programming examples.

Classes

Class	Description
CacheKeyNotFoundException	A CacheKeyNotFoundException exception occurs if a key cannot be found in the cache.
ClassAliasAttribute	Specifies the class alias that you can use to correlate different class names that have the same class alias. The class alias and class fields or types are used to identify a unique class type ID during the object class serialization and de-serialization. The [ClassAlias] annotation can be specified for a user defined class.
ClientServerLoaderException	A ClientServerLoaderException exception is a base exception for any client/server operation exceptions.
DuplicateKeyException	A DuplicateKeyException exception occurs if a key cannot be inserted into the backing map because an object with the same key already exists.
FieldAliasAttribute	Specifies the field alias that you can use to correlate different class field names that have the same class field alias. The class alias and field alias or types are used to identify a unique class type ID during the object class serialization and de-serialization. The [FieldAlias] annotation can be specified for user defined class fields.
LoaderException	A LoaderException exception is the base exception that results for any exceptions that are encountered by a Loader.

[LockDeadlockException](on)	A LockStrategyNotSupportedException exception occurs when the lock manager detects a deadlock. This exception occurs to prevent the deadlock.
[LockException](on)	A LockException exception indicates errors with locking operations.
[LockStrategyNotSupportedException](on)	A LockStrategyNotSupportedException exception occurs if a map is configured with an unsupported lock strategy.
[LockTimeoutException](on)	A LockTimeoutException exception occurs when the lock manager detects that the lock wait time exceeded the maximum wait time. The timeout might be the result of a deadlock. If a deadlock is causing the timeout, the timeout is used to break the deadlock.
[MultiplePartitionWriteException](on)	A MultiplePartitionWriteException exception is a base exception for client/server operations when a user attempts to write to multiple remote partitions on remote servers in the same transaction.
[OptimisticCollisionException](on)	An OptimisticCollisionException occurs when an optimistic locking strategy is used and more than one update transaction collides on the same map entry of an ObjectGrid instance. The first transaction to commit updates the version object for the map entry. Other transactions that read this same map entry before committing have the previous version object. When the other transactions try to commit, the version object that is read does not match the version that was last committed. Therefore, other transactions are prevented from updating a map entry with stale data. The default OptimisticCallback plug-in is used by the run time if an implementation is not provided by the application. If a well-constructed equals(Object) method is not on your value object, this exception occurs because the entire value object is used as the version object.
[StaleMapEntryException](on)	Because this exception indicates that the map entry contains stale data, stale map entries or entries as identified by the key parameter that is passed to the OptimisticCollisionException(String, String, String, Object) method are invalidated. If this exception is thrown by a Loader plug-in and a null reference is used as the key parameter by the loader, the run time assumes that the loader does not know which entry caused the exception. In this scenario, the LogSequence object is passed to the Loader.batchUpdate(TxID, LogSequence) method to determine which map entries to invalidate. Each LogElement entry in the LogSequence object that is type update or

	delete is invalidated.
PartitionKeyAttribute	Specifies one or more attributes to use to calculate the partition hash code. The PartitionKey attribute can be specified for the class using a path syntax to identify a single attribute. Multiple attributes can be specified using additional PartitionKey annotations on each field, using the Order attribute to specify the order in which the hash codes will be calculated. The PartitionKey attribute is not inheritable.
ReadOnlyException	A ReadOnlyException exception occurs when a modify operation is attempted on a read-only map.
TargetNotAvailableException	A TargetNotAvailableException occurs when a remote target was not found or was not reachable.
TransactionAffinityException	A TransactionAffinityException exception occurs during server failover for inflight transactions. Applications can try the transaction again.
UnavailableServiceException	An UnavailableServiceException exception occurs when all servers are not running, or when all services are not available even though servers are running.
UndefinedMapException	An UndefinedMapException exception occurs to indicate that the map that an application tried to access is not defined in the ObjectGrid.

Interfaces

Interface	Description
IGridMap TKey, TValue	The top level interface for all maps. Use the interface to retrieve an appropriate IGridMap instance. Different IGridMap implementations are returned which allow additional operations for specific configurations and usage patterns.
IGridMapPessimisticAutoTx TKey, TValue	This is a handle to a map using automatic transaction demarcation. An instance of this IGridMapPessimisticAutoTx can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance. This is a handle to a map using the pessimistic locking strategy and

	manual transaction demarcation. All data access operations must occur within an active transaction.
IGridMapPessimisticTx TKey, TValue	Use the Transaction property to access the IGridTransaction instance that is associated with this map instance, and use the Begin method to begin a transaction. All keys in a single transaction must resolve to the same partition. An instance of this IGridMapPessimisticTx is not thread-safe, and can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance.
IPartitionManager TKey, TValue	An IPartitionManager provides properties and methods for determining how partitions are calculated. Retrieve an IPartitionManager from an IGridMap TKey, TValue .

Enumerations

Enumeration	Description
LockMode	Specifies the strength of a lock to acquire.
TtlType	Every grid map has an optional, built in timed based evictor that is referred to as the "time to live" evictor or TTL evictor. Each grid map entry has an expiration time that determines how long the entry is allowed to live in the grid map. When the expiration time is reached, the TTL evictor causes the expired entry to be evicted from the grid map. This enum defines the TTLType value constants that determine how the the expiration time is computed for a map entry.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A CacheKeyNotFoundException exception occurs if a key cannot be found in the cache.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map CacheKeyNotFoundException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

CacheKeyNotFoundException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CacheKeyNotFoundException type exposes the following members.

Constructors

Name	Description
CacheKeyNotFoundException	Initializes a new instance of the CacheKeyNotFoundException class.
CacheKeyNotFoundException(Exception)	Initializes a new instance of the CacheKeyNotFoundException class with a specified exception cause.
CacheKeyNotFoundException(String)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message.
CacheKeyNotFoundException(String, Exception)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message and exception cause.

Back to Top

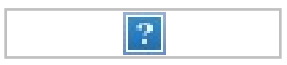
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

CacheKeyNotFoundException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
CacheKeyNotFoundException	Initializes a new instance of the CacheKeyNotFoundException class.
CacheKeyNotFoundException(Exception)	Initializes a new instance of the CacheKeyNotFoundException class with a specified exception cause.
CacheKeyNotFoundException(String)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message.
CacheKeyNotFoundException(String, Exception)	Initializes a new instance of the CacheKeyNotFoundException class with the specified error message and exception cause.

Back to Top

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CacheKeyNotFoundException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

CacheKeyNotFoundException Class
CacheKeyNotFoundException Members
CacheKeyNotFoundException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CacheKeyNotFoundException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

CacheKeyNotFoundException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CacheKeyNotFoundException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

CacheKeyNotFoundException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the class alias that you can use to correlate different class names that have the same class alias. The class alias and class fields or types are used to identify a unique class type ID during the object class serialization and de-serialization.

The [ClassAlias] annotation can be specified for a user defined class.

Inheritance Hierarchy

System Object
  System Attribute
    IBM.WebSphere.Caching.Map ClassAliasAttribute

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

The syntax for the class could look like :

[ClassAlias("ACME_Employee")]

In the above example, ACME_Employee is the class alias for this user defined class.

If [ClassAlias] annotation is not defined, the name of this class is set as the ClassAlias.

Copy to ClipboardPrint

```
[ClassAlias("ACME_Employee")]
class Employee1 {
    [FieldAlias("Employee ID")]
    int empId = -1;

    [FieldAlias("Department No.")]
    int deptId = -1;

    [FieldAlias("Year Salary")]
    float salary = 0;

    [FieldAlias("SEX")]
    String sex = "M";

    int age = -1;
    String homeAddress = "";
```

When a ClassAlias and/or FieldAlias are specified in a user defined class, the ClassAlias and/or FieldAlias will be used to create or correlate with an object that are stored or will be stored in the grid. If two user defined classes (in a separate .NET application environment) have the different class name, but they were marked as the same ClassAlias, and all fields and field types are matched between these 2 classes, they will be correlated with the same class type ID even though they have the different class name. This way will allow the same class metadata to be reused between these 2 classes when running serialization and de-serialization in the different .NET application runtime, as well as to shared with Java when

the Alias for the class defined in Java and fields are also matched.

See Also

ClassAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClassAliasAttribute type exposes the following members.

Constructors

Name	Description
ClassAliasAttribute	The class alias.

Back to Top

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
Alias	The class alias.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

ClassAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The class alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

value
Type: System String
The class alias value.

See Also

ClassAliasAttribute Class
ClassAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClassAliasAttribute type exposes the following members.

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

ClassAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClassAliasAttribute type exposes the following members.

Properties

	Name	Description
	Alias	The class alias.
	TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

ClassAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The class alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

Field Value

The class alias.
See Also

ClassAliasAttribute Class
ClassAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ClientServerLoaderException exception is a base exception for any client/server operation exceptions.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LoaderException
        IBM.WebSphere.Caching.Map ClientServerLoaderException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ClientServerLoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerLoaderException type exposes the following members.

Constructors

Name	Description
ClientServerLoaderException	Initializes a new instance of the ClientServerLoaderException class.
ClientServerLoaderException(Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified exception cause.
ClientServerLoaderException(String)	Initializes a new instance of the ClientServerLoaderException class with the specified error message.
ClientServerLoaderException(String, Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ClientServerLoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ClientServerLoaderException	Initializes a new instance of the ClientServerLoaderException class.
ClientServerLoaderException(Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified exception cause.
ClientServerLoaderException(String)	Initializes a new instance of the ClientServerLoaderException class with the specified error message.
ClientServerLoaderException(String, Exception)	Initializes a new instance of the ClientServerLoaderException class with the specified error message and exception cause.

Back to Top

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ClientServerLoaderException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ClientServerLoaderException Class
ClientServerLoaderException Members
ClientServerLoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerLoaderException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerLoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ClientServerLoaderException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ClientServerLoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A DuplicateKeyException exception occurs if a key cannot be inserted into the backing map because an object with the same key already exists.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map DuplicateKeyException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

DuplicateKeyException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The DuplicateKeyException type exposes the following members.

Constructors

Name	Description
DuplicateKeyException	Initializes a new instance of the DuplicateKeyException class.
DuplicateKeyException(Exception)	Initializes a new instance of the DuplicateKeyException class with a specified exception cause.
DuplicateKeyException(String)	Initializes a new instance of the DuplicateKeyException class with the specified error message.
DuplicateKeyException(String, Exception)	Initializes a new instance of the DuplicateKeyException class with the specified error message and exception cause.

Back to Top

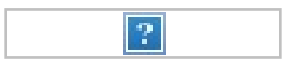
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception.

		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

DuplicateKeyException Class

IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
DuplicateKeyException	Initializes a new instance of the DuplicateKeyException class.
DuplicateKeyException(Exception)	Initializes a new instance of the DuplicateKeyException class with a specified exception cause.
DuplicateKeyException(String)	Initializes a new instance of the DuplicateKeyException class with the specified error message.
DuplicateKeyException(String, Exception)	Initializes a new instance of the DuplicateKeyException class with the specified error message and exception cause.

Back to Top

See Also

DuplicateKeyException Class
DuplicateKeyException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the DuplicateKeyException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies an error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

DuplicateKeyException Class
DuplicateKeyException Members
DuplicateKeyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The DuplicateKeyException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

DuplicateKeyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The DuplicateKeyException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

DuplicateKeyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the field alias that you can use to correlate different class field names that have the same class field alias. The class alias and field alias or types are used to identify a unique class type ID during the object class serialization and de-serialization.

The [FieldAlias] annotation can be specified for user defined class fields.

Inheritance Hierarchy

System Object
  System Attribute
    IBM.WebSphere.Caching.Map FieldAliasAttribute

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

The syntax for the field alias could look as follows :

[FieldAlias("Employee ID")]

In the above example, "Employee ID" is a field alias for the empId field in this user defined class.

If [FieldAlias] annotation is not defined, the name of this field is set as the FieldAlias.

Copy to ClipboardPrint

```
[ClassAlias("ACME_Employee")]
class Employee1 {
    [FieldAlias("Employee ID")]
    int empId = -1;

    [FieldAlias("Department No.")]
    int deptId = -1;

    [FieldAlias("Year Salary")]
    float salary = 0;

    [FieldAlias("SEX")]
    String sex = "M";

    int age = -1;
    String homeAddress = "";
```

When a ClassAlias and/or FieldAlias are specified in a user defined class, the ClassAlias and/or FieldAlias will be used to create or correlate with an object that are stored or will be stored in the grid. If two user defined classes (in a separate .NET application environment) have the different class name, but they were marked as the same ClassAlias, and all fields and field types are matched between these 2 classes, they will be correlated with the same class type ID even though they have the different class name. This way will allow the same class metadata to be reused between these 2 classes when running serialization and de-

serialization in the different .NET application runtime, as well as to shared with Java when the Alias for the class defined in Java and fields are also matched.

See Also

FieldAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The FieldAliasAttribute type exposes the following members.

Constructors

Name	Description
FieldAliasAttribute	The field alias.

Back to Top

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
Alias	The field alias.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

FieldAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The field alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

value
Type: System String
The field alias value.

See Also

FieldAliasAttribute Class
FieldAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The FieldAliasAttribute type exposes the following members.

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

FieldAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The FieldAliasAttribute type exposes the following members.

Properties

	Name	Description
	Alias	The field alias.
	TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

FieldAliasAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The field alias.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The field alias.

See Also

FieldAliasAttribute Class
FieldAliasAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The top level interface for all maps. Use the interface to retrieve an appropriate IGridMap instance.

Different IGridMap implementations are returned which allow additional operations for specific configurations and usage patterns.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
: Generic type key.

TValue
: Generic type value.

See Also

IGridMap TKey, TValue _Members
IBM.WebSphere.Caching.Map Namespace
IBM.WebSphere.Caching IGrid

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMap TKey, TValue type exposes the following members.

Properties

	Name	Description
	Grid	Retrieves the IGrid instance associated with this map.
	Name	Retrieves the map name.
	PartitionManager	Retrieves the IPartitionManager associated with this map.

Back to Top

See Also

IGridMap TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMap TKey, TValue type exposes the following members.

Properties

	Name	Description
	Grid	Retrieves the IGrid instance associated with this map.
	Name	Retrieves the map name.
	PartitionManager	Retrieves the IPartitionManager associated with this map.

Back to Top

See Also

IGridMap TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the IGrid instance associated with this map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The IGrid instance.

See Also

IGridMap TKey, TValue Interface
IGridMap TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the map name.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

Field Value

The map name.
See Also

IGridMap TKey, TValue Interface
IGridMap TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the IPartitionManager associated with this map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The IPartitionManager associated with this map.

See Also

IGridMap TKey, TValue Interface
IGridMap TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

This is a handle to a map using automatic transaction demarcation.

An instance of this IGridMapPessimisticAutoTx can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
: Generic type key.

TValue
: Generic type value.

Remarks

Each data access method includes a "Specification details" table that includes the following information:

Required permission:	The permission required to use the API.
Pessimistic read lock acquired:	The type of lock that is acquired when using pessimistic locking with repeatable read transaction isolation.
Cache tier:	Identifies the map cache tiers that are included when fetching or updating cache entries in the call and under what circumstances. The following tiers are available for IGridMapPessimisticAutoTx maps: • Server Cache (data grid) • Loader (if enabled)

Examples

This sample demonstrates how to put a new cache entry into the data grid:
Copy to ClipboardPrint

```
// Assume we have already connected to the Grid...
IGrid grid = ...

// Retrieve a new map instance.
IGridMapPessimisticAutoTx<long, string> map = grid.GetGridMapPessimisticAutoTx<long, string>("MyP
essimisticMap");

try
{
    // Put the entry in the cache.
    map.Put(123, "Value to cache");
```

```
}
catch(GridException)
{
    // Handle any concequences of failed put.
}

// Dispose the map (optional, but it improves performance)
map.Dispose();
```

See Also

IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticAutoTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered, which may be during a flush or commit operation, in which the exception will be an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a nullkey cannot be specified.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values associated with the list of keys that are specified in the keyList. If the value is not found, a null is returned. If the map supports null values, use the ContainsKeyAll(IList TKey) to test for multiple keys that may have a null value.
Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding a new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method

	in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException is thrown during commit.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException is thrown.
TouchAll	Updates the last access time for the data grid entries specified in the keyList without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
TryAdd	A variant of Add(TKey, TValue) that does not throw exceptions.
TryAddAll	Adds the key-value pairs to the data grid.
TryInvalidate	Removes the entry associated with the specified key from the cache, leaving the data behind unchanged.
TryInvalidateAll	Removes the entries associated with the keys that are specified in the keyList from the cache, leaving the data behind unchanged.
TryPut	Adds the key-value pair to the grid. If an entry for the key exists in the data grid, the value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryPutAll	Adds each key-value pair to the data grid. If an entry for a key exists in the data grid, its value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryRemove	Removes the entry that is associated with the specified key from the data grid. If the key has no matching entry in the map, no action is taken.
TryRemoveAll	Removes the list of entries associated with the keys that are specified in the keyList. If a key in the keyList has no matching entry in the map, no action is taken for that key.
TryReplace	Replaces the value of a key-value pair in the data grid. If an entry for the key exists in the data grid, its value is replaced with the specified value.

TryReplaceAll	Replaces each key-value pair that is specified in the dictionary object in the data grid. If an entry for the key exists in the data grid, its value is replaced with the corresponding value specified.
TryTouch	Updates the last access time for the data grid entry that matches the key.
TryTouchAll	Updates the last access time for the data grid entries that match each key in the keyList.

Back to Top

Properties

Name	Description
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	Gets or sets the value in the map using the specified key.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
MapName	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
PartitionManager	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever".

Live	This property can only be used when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException is thrown. To disable the TTL timeout, use a value of TimeSpan.Zero. To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.
TtlEvictorType	Gets the TtlType of the map's TTL evictor.

Back to Top
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticAutoTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException is thrown when the duplicate key is discovered, which may be during a flush or commit operation, in which the exception will be an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a nullkey cannot be specified.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values associated with the list of keys that are specified in the keyList. If the value is not found, a null is returned. If the map supports null values, use the ContainsKeyAll(IList TKey) to test for multiple keys that may have a null value.
Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding a new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method

	in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, it will be ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, it will be ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException is thrown during commit.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException is thrown.
TouchAll	Updates the last access time for the data grid entries specified in the keyList without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException is thrown.
TryAdd	A variant of Add(TKey, TValue) that does not throw exceptions.
TryAddAll	Adds the key-value pairs to the data grid.
TryInvalidate	Removes the entry associated with the specified key from the cache, leaving the data behind unchanged.
TryInvalidateAll	Removes the entries associated with the keys that are specified in the keyList from the cache, leaving the data behind unchanged.
TryPut	Adds the key-value pair to the grid. If an entry for the key exists in the data grid, the value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryPutAll	Adds each key-value pair to the data grid. If an entry for a key exists in the data grid, its value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.
TryRemove	Removes the entry that is associated with the specified key from the data grid. If the key has no matching entry in the map, no action is taken.
TryRemoveAll	Removes the list of entries associated with the keys that are specified in the keyList. If a key in the keyList has no matching entry in the map, no action is taken for that key.
TryReplace	Replaces the value of a key-value pair in the data grid. If an entry for the key exists in the data grid, its value is replaced with the specified value.

TryReplaceAll	Replaces each key-value pair that is specified in the dictionary object in the data grid. If an entry for the key exists in the data grid, its value is replaced with the corresponding value specified.
TryTouch	Updates the last access time for the data grid entry that matches the key.
TryTouchAll	Updates the last access time for the data grid entries that match each key in the keyList.

Back to Top

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface

IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pair to the data grid.

The key must not exist before executing this method.

A DuplicateKeyException is thrown when the duplicate key is discovered.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be inserted in the data grid.

value
Type: TValue
Specifies the value to be inserted in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when the key exists.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INSERT
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds multiple key-value pairs to the data grid.

The keys must not exist before executing this method.

A DuplicateKeyException is thrown when the duplicate key is discovered, which may be during a flush or commit operation, in which the exception will be an inner exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to be inserted into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullentries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when a dictionary key in entries already exists in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission: MapPermission.INSERT	
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a nullkey cannot be specified.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to test in the map.

Return Value

Returns true if the key is found, false otherwise.
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys to test in the map.

Return Value

Specifies a list of boolean values. If the key is found in the keyList, true is listed. Otherwise, false is returned.
Exceptions

Exception	Condition
System ArgumentException	Occurs when keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the value that is associated with the specified key. If the value is not found, a null is returned.

If the map supports null values, use the ContainsKey(TKey) to test for a key that may have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to query.

Return Value

The value that is associated with the specified key if it exists; otherwise null is returned.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified, or when the data type of the value that was obtained from the data grid does not match the declared data type.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the values associated with the list of keys that are specified in the keyList. If the value is not found, a null is returned.

If the map supports null values, use the ContainsKeyAll(IList TKey) to test for multiple keys that may have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to query.

Return Value

A list of values that are associated with the supplied keys. If the value associated with a particular key is not in the data grid, null is returned in the list at the position that is associated with the key.
Exceptions

Exception	Condition
System ArgumentException	Occurs when the keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing,
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store).

If the key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store).

If a key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be put in the data grid.

value
Type: TValue
Specifies the value to be put in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts multiple key-value pairs to the data grid, replacing or adding a new entries to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to be put into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullentries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store).

If the key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store).

If a key cannot be found in the map, it will be ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store).

If the key cannot be found in the data grid a CacheKeyNotFoundException is thrown during commit.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be updated.

value
Type: TValue
Specifies the value to be updated in the data grid and Loader.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store).

If a key cannot be found in the map a CacheKeyNotFoundException is thrown.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to replace in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within entries.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs when the key is not found in any of the cache tiers or Loader. This may be deferred until commit time.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Resets the configurable settings for the map back to configured values.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

This method only resets configuration parameters that can be overridden by the client.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value.

If the key cannot be found in the map a CacheKeyNotFoundException is thrown.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs if the key does not exist in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Updates the last access time for the data grid entries specified in the keyList without locking the entries or fetching the values.

If a key cannot be found in the map a CacheKeyNotFoundException is thrown.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the keys to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs if the key does not exist in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A variant of Add(TKey, TValue) that does not throw exceptions.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be inserted in the data grid.

value
Type: TValue
Specifies the value to be inserted in the data grid.

Return Value

Returns true if successful, false if unsuccessful.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pairs to the data grid.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object of key-value pairs to be inserted into the data grid.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If entries is null or a null key is specified in a dictionary entry or a dictionary key in entries exists, false is returned. If all keys in the dictionary entries do not resolve to the same partition or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Removes the entry associated with the specified key from the cache, leaving the data behind unchanged.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for which you want to remove the value.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If key is null or the caller does not have authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Removes the entries associated with the keys that are specified in the keyList from the cache, leaving the data behind unchanged.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
The list of keys whose values are to be removed.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If keyList is null or a null key is specified in a keyList entry, false is returned. If all keys in the keyList do not resolve to the same partition or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pair to the grid. If an entry for the key exists in the data grid, the value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
: Type: TKey
specifies the key to be inserted into the data grid.

value
: Type: TValue
specifies the value to be inserted into the data grid.

Return Value

Returns true if successful, false if unsuccessful.

Remarks

If the key is null, or the caller has insufficient authority to run this operation, false is returned.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds each key-value pair to the data grid. If an entry for a key exists in the data grid, its value is updated with the specified value. If an entry for a key does not exist in the data grid, the key-value pair is added to the data grid.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object that contains the key-value pairs to be added to the data grid.

Return Value

Returns true if successful, false if unsuccessful.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid. If the key has no matching entry in the map, no action is taken.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for which you want to remove the entry.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If key is null or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Removes the list of entries associated with the keys that are specified in the keyList. If a key in the keyList has no matching entry in the map, no action is taken for that key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
The list of keys whose entries are to be removed.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If keyList is null or a null key is specified in a keyList entry, false is returned. If all keys in the keyList do not resolve to the same partition or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Replaces the value of a key-value pair in the data grid. If an entry for the key exists in the data grid, its value is replaced with the specified value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for the entry to be replaced.

value
Type: TValue
Specifies the new value.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If key is null or the key does not match an existing entry in the data grid, false is returned. If the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Replaces each key-value pair that is specified in the dictionary object in the data grid. If an entry for the key exists in the data grid, its value is replaced with the corresponding value specified.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: System.Collections.Generic IDictionary TKey, TValue
Specifies a dictionary object that contains key-value pairs to be replaced in the data grid.

Return Value

Returns true if successful, false if unsuccessful.
Remarks

If entries is null, a null key is specified in a dictionary entry, or the key does not match an existing entry in the data grid, false is returned. If all keys in the dictionary entries do not resolve to the same partition, or the caller has insufficient authority to run this operation, false is returned.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Updates the last access time for the data grid entry that matches the key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to have last access time updated.

Return Value

Returns true if successful, false if unsuccessful.

Remarks

If the key is null or the key does not match an existing entry in the data grid, false is returned.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Updates the last access time for the data grid entries that match each key in the keyList.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys to have last access time updated.

Return Value

Returns true if successful, false if unsuccessful.

Remarks

If an entry is null or any key in the keyList is null, false is returned. If a key in the keyList does not match an existing entry in the data grid or all keys in the keyList do not resolve to the same partition, false is returned.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticAutoTx TKey, TValue type exposes the following members.

Properties

Name	Description
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	Gets or sets the value in the map using the specified key.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
MapName	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
PartitionManager	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever". This property can only be used when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException is thrown.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

TtlEvictorType

Gets the TtlType of the map's TTL evictor.

Back to Top
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the value in the map using the specified key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
The key of the value to set or get.

Field Value

The value associated with key to get or set.

Remarks

This indexer retreives or put's the key and value into the map using the Get(TKey) and Put(TKey, TValue) methods respectively.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace
IGridMapPessimisticAutoTx TKey, TValue Get(TKey)
IGridMapPessimisticAutoTx TKey, TValue Put(TKey, TValue)

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The maximum time to wait when acquiring a lock on an item in the grid map.

Exceptions

Exception	Condition
System ArgumentException	Occurs if the value is not >= Zero.

Remarks

To prevent deadlocks from occurring, the grid map has a default timeout value of 15 seconds; however, on a heavily loaded system, lock timeouts can occur without an actual deadlock. Use this property to increase the value from the default to prevent false lock timeout exceptions from occurring.

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever".

This property can only be used when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException is thrown.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Exceptions

Exception	Condition
System ArgumentException	Occurs if the TimeSpan is not >=0 or TimeSpan.MinValue or if the TtlEvictorType property is not LastAccessTime or LastUpdateTime.

Remarks

Required Permission: MapPermission.INVALIDATE

See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the TtlType of the map's TTL evictor.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The TtlType of the map's TTL evictor.
See Also

IGridMapPessimisticAutoTx TKey, TValue Interface
IGridMapPessimisticAutoTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

This is a handle to a map using the pessimistic locking strategy and manual transaction demarcation. All data access operations must occur within an active transaction.

Use the Transaction property to access the IGridTransaction instance that is associated with this map instance, and use the Begin method to begin a transaction. All keys in a single transaction must resolve to the same partition.

An instance of this IGridMapPessimisticTx is not thread-safe, and can only be used by the thread at a time. Use the Dispose method when finished with the map, to improve performance.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
Generic type key.

TValue
Generic type value.

Remarks

Each data access method includes a "Specification details" table that includes the following information:

Required permission:	The permission required to use the API.
Pessimistic read lock acquired:	The type of lock that is acquired when you are using pessimistic locking with repeatable read transaction isolation.
Pessimistic read lock held:	The type of lock that is held for the duration of the transaction with repeatable read transaction isolation. Locks can be upgraded but not demoted during a transaction.
Cache tier:	Identifies the map cache tiers that are included when fetching or updating cache entries in the call and under what circumstances. The following tiers are available for IGridMapPessimisticTx maps: • Transactional Cache • Server Cache (data grid) • Loader (if enabled)

Examples

This sample demonstrates how to put a new cache entry into the data grid:
Copy to ClipboardPrint

```
// Assume we have already connected to the Grid...
IGrid grid = ...

// Retrieve a new map instance.
IGridMapPessimisticTx<long, string> map = grid.GetGridMapPessimisticTx<long, string>("MyPessimist
icMap");

// Start a transaction.
map.Transaction.Begin();

try
{
    // Lock the entry in the data grid with an Upgradable lock.
    map.Lock(123, LockMode.Upgradable);

    // Put the entry in the transactional cache.
    map.Put(123, "Value to cache");

    // Commit the transaction to the data grid.
    map.Transaction.Commit();
}
catch(GridException)
{
    // Clean-up the transaction if the lock could not be
    // acquired, or the commit failed.
    if(map.Transaction.Active)
    {
        try
        {
            map.Transaction.Rollback();
        }
        catch(Exception)
        {
            // Optionally log this exception, or ignore the exception.
        }
    }

    // Dispose the map (optional, but it improves performance)
    map.Dispose();

    // Rethrow the real exception.
    throw;
}
```

See Also

IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before you run this method. A DuplicateKeyException exception results when the duplicate key is discovered. Duplicate keys might be discovered with a flush or commit operation. In this scenario, the exception is an inner exception.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException exception results when the duplicate key is discovered. This discovery might happen during a flush or commit operation. In this scenario, the exception is an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a null key cannot be specified. This API does not hold any locks. Use the Lock(TKey, LockMode) to test for a key and retain a lock.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList. This API does not hold any locks. Use the LockAll(IList TKey , LockMode) to test for a key and retain a lock.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values that are associated with the list of keys that are specified in the keyList. If the value is not found, a null value is returned. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for multiple keys that might have a null value.
GetAndLock	Locks the specified key and retrieves the associated value. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that might have a null value.
GetAndLockAll	Locks the specified keys and retrieves the associated values. If a value is not found, a null is returned in the list. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for a key that might have a null value.

Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Lock	Locks the specified key and tests to see if the key was previously present in the data grid or Loader.
LockAll	Locks the specified keys and tests to see if each was previously present in the data grid or Loader.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException exception results during the commit operation.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException exception results.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.
TouchAll	Updates the last access time for the data grid entries that are specified in the keyList list without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Back to Top

Properties

Name	Description

Name	
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	An indexer that retreives or puts the key and value into the map with the Get(TKey) and Put(TKey, TValue) methods.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
Name	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
Partition	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
PartitionManager	
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever". This property can be used only when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException exception results. To disable the TTL timeout, use a value of TimeSpan.Zero. To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.
Transact	The Transaction instance used to configure and demarcate a transaction. (Inherited from ITransactionable.)

ionTtlEvictorType Retrieves the time to live type for the evictor on the map.

Back to Top
See Also

IGridMapPessimisticTx TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticTx TKey, TValue type exposes the following members.

Methods

Name	Description
Add	Adds the key-value pair to the data grid. The key must not exist before you run this method. A DuplicateKeyException exception results when the duplicate key is discovered. Duplicate keys might be discovered with a flush or commit operation. In this scenario, the exception is an inner exception.
AddAll	Adds multiple key-value pairs to the data grid. The keys must not exist before executing this method. A DuplicateKeyException exception results when the duplicate key is discovered. This discovery might happen during a flush or commit operation. In this scenario, the exception is an inner exception.
ContainsKey	Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a null key cannot be specified. This API does not hold any locks. Use the Lock(TKey, LockMode) to test for a key and retain a lock.
ContainsKeyAll	Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList. This API does not hold any locks. Use the LockAll(IList TKey , LockMode) to test for a key and retain a lock.
Dispose	Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)
Get	Retrieves the value that is associated with the specified key. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that may have a null value.
GetAll	Retrieves the values that are associated with the list of keys that are specified in the keyList. If the value is not found, a null value is returned. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for multiple keys that might have a null value.
GetAndLock	Locks the specified key and retrieves the associated value. If the value is not found, a null is returned. If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that might have a null value.
GetAndLockAll	Locks the specified keys and retrieves the associated values. If a value is not found, a null is returned in the list. If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for a key that might have a null value.

Invalidate	Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
InvalidateAll	Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Lock	Locks the specified key and tests to see if the key was previously present in the data grid or Loader.
LockAll	Locks the specified keys and tests to see if each was previously present in the data grid or Loader.
Put	Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.
PutAll	Puts multiple key-value pairs to the data grid, replacing or adding new entries to each data grid tier as needed. Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.
Remove	Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store). If the key cannot be found in the map, the operation is ignored.
RemoveAll	Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store). If a key cannot be found in the map, the operation is ignored.
Replace	Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store). If the key cannot be found in the data grid a CacheKeyNotFoundException exception results during the commit operation.
ReplaceAll	Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store). If a key cannot be found in the map a CacheKeyNotFoundException exception results.
ResetToDefaults	Resets the configurable settings for the map back to configured values.
Touch	Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value. If the key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.
TouchAll	Updates the last access time for the data grid entries that are specified in the keyList list without locking the entries or fetching the values. If a key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Back to Top

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds the key-value pair to the data grid.

The key must not exist before you run this method.

A DuplicateKeyException exception results when the duplicate key is discovered. Duplicate keys might be discovered with a flush or commit operation. In this scenario, the exception is an inner exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be inserted in the data grid.

value
Type: TValue
Specifies the value to be inserted in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullkey is specified.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when the key exists.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INSERT
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Adds multiple key-value pairs to the data grid.

The keys must not exist before executing this method.

A DuplicateKeyException exception results when the duplicate key is discovered. This discovery might happen during a flush or commit operation. In this scenario, the exception is an inner exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
: Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a IOrderedDictionary TKey, TValue object of key-value pairs to be inserted into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a nullentries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching.Map DuplicateKeyException	Occurs when a dictionary key in entries already exists in the map.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INSERT
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Looks in the map for an entry that is associated with the specified the key. If an entry is found, true is returned. If an entry is not found, false is returned. Data grids do not support null key values, so a null key cannot be specified.

This API does not hold any locks. Use the Lock(TKey, LockMode) to test for a key and retain a lock.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to test in the map.

Return Value

Returns true if the key is found, false otherwise.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	No
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Looks in the map for the entries that are associated with the specified keys in the keyList. If an entry is located, true is returned. If an entry is not located, false is returned. Data grids do not support null key values, so a null key cannot be specified in a keyList.

This API does not hold any locks. Use the LockAll(IList TKey , LockMode) to test for a key and retain a lock.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys to test in the map.

Return Value

Specifies a list of boolean values. If the key is found in the keyList, true is listed. Otherwise, false is returned.
Exceptions

Exception	Condition
System ArgumentException	Occurs when keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	No
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the value that is associated with the specified key. If the value is not found, a null is returned.

If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that may have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to query.

Return Value

The value that is associated with the specified key if it exists; otherwise null is returned.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified, or when the data type of the value that was obtained from the data grid does not match the declared data type.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the values that are associated with the list of keys that are specified in the keyList. If the value is not found, a null value is returned.

If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for multiple keys that might have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to query.

Return Value

A list of values that are associated with the supplied keys. If the value associated with a particular key is not in the data grid, null is returned in the list at the position that is associated with the key.

Exceptions

Exception	Condition
System ArgumentException	Occurs when the keyList is null, or when a null key is specified in keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing,
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared
Pessimistic locks held:	Yes
Cache tier:	For each key, progresses to all tiers until the key is found.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Locks the specified key and retrieves the associated value. If the value is not found, a null is returned.

If the map supports null values, use the Lock(TKey, LockMode) or the ContainsKey(TKey) to test for a key that might have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

Specifies the value that is associated with the specified key if it exists. Otherwise, null is returned.
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Locks the specified keys and retrieves the associated values. If a value is not found, a null is returned in the list.

If the map supports null values, use the LockAll(IList TKey , LockMode) or the ContainsKeyAll(IList TKey) to test for a key that might have a null value.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

A list of values that are associated with the supplied keys. If the value associated with a particular key is not in the data grid, null is returned in the list at the position that is associated with the key.

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the keys are found and the appropriate locks are acquired.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid, without affecting the Loader (back-end persistent store).

If the key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entries that are associated with the specified keyList from the data grid, without affecting the Loader (back-end persistent store).

If a key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be invalidated from the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.INVALIDATE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Locks the specified key and tests to see if the key was previously present in the data grid or Loader.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

Returns true if the key is found in the data grid or Loader (back-end persistent store).
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Locks the specified keys and tests to see if each was previously present in the data grid or Loader.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to lock.

lockMode
Type: IBM.WebSphere.Caching.Map LockMode
Specifies the type of lock to acquire.

Return Value

A list of bool that are associated with the supplied keys, where true indicates that the key was found in the data grid or Loader (back-end persistent store).
Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.READ
Pessimistic locks acquired:	LockMode.Shared, LockMode.Upgradable or LockMode.Exclusive
Pessimistic locks held:	Yes
Cache tier:	Progresses to all tiers until the key is found and the appropriate lock is acquired.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts the specified key and value into the data grid, replacing or adding a new entry to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsert method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
: Type: TKey
Specifies the key to be put in the data grid.

value
: Type: TValue
Specifies the value to be put in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Puts multiple key-value pairs to the data grid, replacing or adding new entries to each data grid tier as needed.

Note: This method has the same specification as the ObjectMap.upsertAll method in the eXtreme Scale Java client.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a IOrderedDictionary TKey, TValue object of key-value pairs to be put into the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null entries is specified, or when a null key is specified in entries.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entry that is associated with the specified key from the data grid and Loader (back-end persistent store).

If the key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Removes the entries that are associated with the specified keyList from the data grid and Loader (back-end persistent store).

If a key cannot be found in the map, the operation is ignored.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the list of keys to be removed from the data grid and Loader

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList..
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.REMOVE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Replaces the existing entry that is associated with the specified key with the specified value from the data grid and Loader (back-end persistent store).

If the key cannot be found in the data grid a CacheKeyNotFoundException exception results during the commit operation.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to be updated.

value
Type: TValue
Specifies the value to be updated in the data grid and Loader.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Replaces multiple key-value pairs into the data grid and Loader (back-end persistent store).

If a key cannot be found in the map a CacheKeyNotFoundException exception results.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

entries
Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a IOrderedDictionary TKey, TValue object of key-value pairs to replace in the data grid.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within entries.
IBM.WebSphere.Caching Grid Exception	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Map CacheKeyNotFoundException	Occurs when the key is not found in any of the cache tiers or Loader. This exception might be deferred until commit time.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

Remarks

Specification details:

Required client permission:	MapPermission.WRITE
Cache tier:	Applied to all tiers during commit.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Resets the configurable settings for the map back to configured values.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Remarks

This method resets configuration parameters that can be overridden by the client only.

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Updates the last access time for the data grid entry that matches the key without locking the entry or fetching the value.

If the key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null key is specified.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccesControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Updates the last access time for the data grid entries that are specified in the keyList list without locking the entries or fetching the values.

If a key cannot be found in the map a CacheKeyNotFoundException exception results during the commit operation.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies the keys to have last access time updated.

Return Value

Exceptions

Exception	Condition
System ArgumentException	Occurs when a null keyList is specified, or when a null key is specified within keyList.
IBM.WebSphere.Caching GridException	Occurs when an error occurs during processing.
IBM.WebSphere.Caching.Security AccessControlException	Occurs when the caller has insufficient authority to perform this operation.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IGridMapPessimisticTx TKey, TValue type exposes the following members.

Properties

Name	Description
Grid	Retrieves the IGrid instance associated with this map. (Inherited from IGridMap TKey, TValue .)
Item	An indexer that retreives or puts the key and value into the map with the Get(TKey) and Put(TKey, TValue) methods.
LockTimeout	Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.
Name	Retrieves the map name. (Inherited from IGridMap TKey, TValue .)
PartitionManager	Retrieves the IPartitionManager associated with this map. (Inherited from IGridMap TKey, TValue .)
TimeToLive	Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever". This property can be used only when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException exception results.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

Transaction	The Transaction instance used to configure and demarcate a transaction. (Inherited from ITransactionable.)
TtlEvictor Type	Retrieves the time to live type for the evictor on the map.

Back to Top

See Also

IGridMapPessimisticTx TKey, TValue _Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An indexer that retreives or puts the key and value into the map with the Get(TKey) and Put(TKey, TValue) methods.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey

See Also

IGridMapPessimisticTx TKey, TValue_Interface
IGridMapPessimisticTx TKey, TValue_Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the maximum time to wait when acquiring a lock on an item in the grid map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Field Value

The maximum time to wait when acquiring a lock on an item in the grid map.

Exceptions

Exception	Condition
System ArgumentException	Occurs if the value is not >= Zero.

Remarks

To prevent deadlocks from occurring, the grid map has a default timeout value of 15 seconds; however, on a heavily loaded system, lock timeouts can occur without an actual deadlock. Use this property to increase the value from the default to prevent false lock timeout exceptions from occurring.

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Establishes the interval of time that any given cache entry can live for, which is referred to as "time to live" or TTL. Setting a new TTL value affects cache entries that are accessed after this method call occurs. It does not affect any cache entry that was created or accessed prior to this method call. By calling this method on this IGridMapPessimisticTx, any previous value set by the IGridMapPessimisticTx.TimeToLive property is overridden for this map instnace. If this method is never called on the map, the default setting is used. The default setting is to retain the time-to-live value for any existing map entry and to use the default value from map configuration setting if a new map entry is being created. If TTL is never set on the map configuration, the cache entry can live "forever".

This property can be used only when the TtlEvictorType property is set to LastAccessTime or LastUpdateTime on the map configuration. If this method is called on the IGridMapPessimisticTx and the TtlEvictorType is something other than LastAccessTime or LastUpdateTime, an ArgumentException exception results.

To disable the TTL timeout, use a value of TimeSpan.Zero.

To revert the TTL value to the configured default, use a value of TimeSpan.MinValue.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Exceptions

Exception	Condition
System ArgumentException	Occurs if the TimeSpan is not >=0 or TimeSpan.MinValue or if the TtlEvictorType property is not LastAccessTime or LastUpdateTime.

Remarks

Required Permission: MapPermission.INVALIDATE

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the time to live type for the evictor on the map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IGridMapPessimisticTx TKey, TValue Interface
IGridMapPessimisticTx TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An IPartitionManager provides properties and methods for determining how partitions are calculated. Retrieve an IPartitionManager from an IGridMap TKey, TValue .

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Type Parameters

TKey
: Specifies a generic type key.

TValue
: Specifies a generic type value.

Remarks

The partition id is calculated as follows:

- The key class is examined for the PartitionKeyAttribute attribute. If present, the GetHashCode of the referenced attributes is used to calculate the partition.
- Otherwise, the key's GetHashCode method is used to calculate the partition id.

See Also

IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IPartitionManager TKey, TValue type exposes the following members.

Methods

Name	Description
GetPartition	Returns the partition number for the specified map element key.
GetPartitionAll(IList TKey)	Returns a map of partition numbers to map keys from a collection of map element keys.
GetPartitionAll(IOrderedDictionary TKey, TValue)	Returns a map of partition numbers to map keys from a collection of map element keys.

Back to Top

Properties

Name	Description
NumPartitions	Retrieves the number of partitions that are defined for the map.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IPartitionManager TKey, TValue type exposes the following members.

Methods

Name	Description
GetPartition	Returns the partition number for the specified map element key.
GetPartitionAll(IList TKey)	Returns a map of partition numbers to map keys from a collection of map element keys.
GetPartitionAll(IOrderedDictionary TKey, TValue)	Returns a map of partition numbers to map keys from a collection of map element keys.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns the partition number for the specified map element key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: TKey
Specifies the key for which to return a partition number.

Return Value

Returns the partition number for the specified map element key.

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GetPartitionAll(IList TKey)	Returns a map of partition numbers to map keys from a collection of map element keys.
GetPartitionAll(IOrderedDictionary TKey, TValue)	Returns a map of partition numbers to map keys from a collection of map element keys.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map of partition numbers to map keys from a collection of map element keys.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

keyList
Type: System.Collections.Generic IList TKey
Specifies a list of keys for which you want to return partition numbers.

Return Value

Returns a collection of partition numbers and associated keys.

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
GetPartitionAll Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns a map of partition numbers to map keys from a collection of map element keys.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

Parameters

entries
: Type: IBM.WebSphere.Caching IOrderedDictionary TKey, TValue
Specifies a dictionary object of key-value pairs for which you want to return partition numbers.

Return Value

Specifies a collection of partition numbers and associated key-value pairs.

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
GetPartitionAll Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IPartitionManager TKey, TValue type exposes the following members.

Properties

	Name	Description
	NumPartitions	Retrieves the number of partitions that are defined for the map.

Back to Top

See Also

IPartitionManager TKey, TValue Interface
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Retrieves the number of partitions that are defined for the map.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

IPartitionManager TKey, TValue Interface
IPartitionManager TKey, TValue Members
IBM.WebSphere.Caching.Map Namespace

LoaderException Class IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LoaderException exception is the base exception that results for any exceptions that are encountered by a Loader.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LoaderException
        IBM.WebSphere.Caching.Map ClientServerLoaderException
        IBM.WebSphere.Caching.Map UnavailableServiceException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LoaderException type exposes the following members.

Constructors

Name	Description
LoaderException	Initializes a new instance of the LoaderException class.
LoaderException(Exception)	Initializes a new instance of the LoaderException class with a specified exception cause.
LoaderException(String)	Initializes a new instance of the LoaderException class with the specified error message.
LoaderException(String, Exception)	Initializes a new instance of the LoaderException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LoaderException	Initializes a new instance of the LoaderException class.
LoaderException(Exception)	Initializes a new instance of the LoaderException class with a specified exception cause.
LoaderException(String)	Initializes a new instance of the LoaderException class with the specified error message.
LoaderException(String, Exception)	Initializes a new instance of the LoaderException class with the specified error message and exception cause.

Back to Top

See Also

LoaderException Class
LoaderException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
: Specifies the error message that explains the reason for the exception.

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LoaderException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LoaderException Class
LoaderException Members
LoaderException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LoaderException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LoaderException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LoaderException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockStrategyNotSupportedException exception occurs when the lock manager detects a deadlock. This exception occurs to prevent the deadlock.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockException
        IBM.WebSphere.Caching.Map LockTimeoutException
          IBM.WebSphere.Caching.Map LockDeadlockException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

Typically, a deadlock is the result of the following scenario: One transaction gets a map entry, resulting in a weaker lock than an existing lock on the same entry. At commit time, the transaction attempts to promote the weaker lock to a stronger lock to apply the changes to the data store. For example, two transactions try to promote from shared locks to exclusive locks, but each transaction already owns a shared lock.
See Also

LockDeadlockException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockDeadlockException type exposes the following members.

Constructors

Name	Description
LockDeadlockException	Initializes a new instance of the LockDeadlockException class.
LockDeadlockException(Exception)	Initializes a new instance of the LockDeadlockException class with the specified exception cause.
LockDeadlockException(String)	Initializes a new instance of the LockDeadlockException class with the specified error message.
LockDeadlockException(String, Exception)	Initializes a new instance of the LockDeadlockException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor. (Inherited from LockTimeoutException.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

	(Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred. (Inherited from LockTimeoutException.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockDeadlockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockDeadlockException	Initializes a new instance of the LockDeadlockException class.
LockDeadlockException(Exception)	Initializes a new instance of the LockDeadlockException class with the specified exception cause.
LockDeadlockException(String)	Initializes a new instance of the LockDeadlockException class with the specified error message.
LockDeadlockException(String, Exception)	Initializes a new instance of the LockDeadlockException class with the specified error message and exception cause.

Back to Top

See Also

LockDeadlockException Class
LockDeadlockException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockDeadlockException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockDeadlockException Class
LockDeadlockException Members
LockDeadlockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockDeadlockException type exposes the following members.

Methods

	Name	Description
	Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
	Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
	GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
	getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor. (Inherited from LockTimeoutException.)
	GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LockDeadlockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockDeadlockException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred. (Inherited from LockTimeoutException.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LockDeadlockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

LockException Class IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockException exception indicates errors with locking operations.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockException
        IBM.WebSphere.Caching.Map LockTimeoutException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockException Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET API Specification. For additional information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockException type exposes the following members.

Constructors

Name	Description
LockException	Initializes a new instance of the LockException class.
LockException(Exception)	Initializes a new instance of the LockException class with a specified exception cause.
LockException(String)	Initializes a new instance of the LockException class with the specified error message.
LockException(String, Exception)	Initializes a new instance of the LockException class with the specified error message and exception cause.

Back to Top

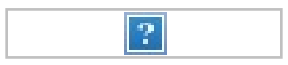
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockException	Initializes a new instance of the LockException class.
LockException(Exception)	Initializes a new instance of the LockException class with a specified exception cause.
LockException(String)	Initializes a new instance of the LockException class with the specified error message.
LockException(String, Exception)	Initializes a new instance of the LockException class with the specified error message and exception cause.

Back to Top

See Also

LockException Class
LockException Members
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockException Class
LockException Members
LockException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockException type exposes the following members.
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top
See Also

LockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockException type exposes the following members.
Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Specifies the strength of a lock to acquire.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Members

Member name	Value	Description
Shared	0	Shared lock that can be obtained if no one is currently holding an exclusive lock.
Upgradable	1	Upgradable lock that can be obtained if no one is currently holding an upgradable or exclusive lock.
Exclusive	2	Exclusive lock that can only be obtained if no locks are held.

See Also

IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockStrategyNotSupportedException exception occurs if a map is configured with an unsupported lock strategy.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockStrategyNotSupportedException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockStrategyNotSupportedException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockStrategyNotSupportedException type exposes the following members.

Constructors

Name	Description
LockStrategyNotSupportedException	Initializes a new instance of the LockStrategyNotSupportedException class.
LockStrategyNotSupportedException(Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with a specified exception cause.
LockStrategyNotSupportedException(String)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message.
LockStrategyNotSupportedException(String, Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockStrategyNotSupportedException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockStrategyNotSupportedException	Initializes a new instance of the LockStrategyNotSupportedException class.
LockStrategyNotSupportedException(Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with a specified exception cause.
LockStrategyNotSupportedException(String)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message.
LockStrategyNotSupportedException(String, Exception)	Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message and exception cause.

Back to Top

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockStrategyNotSupportedException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockStrategyNotSupportedException Class
LockStrategyNotSupportedException Members
LockStrategyNotSupportedException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockStrategyNotSupportedException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LockStrategyNotSupportedException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockStrategyNotSupportedException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LockStrategyNotSupportedException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A LockTimeoutException exception occurs when the lock manager detects that the lock wait time exceeded the maximum wait time. The timeout might be the result of a deadlock. If a deadlock is causing the timeout, the timeout is used to break the deadlock.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LockException
        IBM.WebSphere.Caching.Map LockTimeoutException
          IBM.WebSphere.Caching.Map LockDeadlockException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockTimeoutException type exposes the following members.

Constructors

Name	Description
LockTimeoutException	Initializes a new instance of the LockTimeoutException class.
LockTimeoutException(Exception)	Initializes a new instance of the LockTimeoutException class with a specified exception cause.
LockTimeoutException(String)	Initializes a new instance of the LockTimeoutException class with the specified error message.
LockTimeoutException(String, Exception)	Initializes a new instance of the LockTimeoutException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred.
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

LockTimeoutException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
LockTimeoutException	Initializes a new instance of the LockTimeoutException class.
LockTimeoutException(Exception)	Initializes a new instance of the LockTimeoutException class with a specified exception cause.
LockTimeoutException(String)	Initializes a new instance of the LockTimeoutException class with the specified error message.
LockTimeoutException(String, Exception)	Initializes a new instance of the LockTimeoutException class with the specified error message and exception cause.

Back to Top

See Also

LockTimeoutException Class
LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
: Specifies the error message that explains the reason for the exception.

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the LockTimeoutException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

LockTimeoutException Class
LockTimeoutException Members
LockTimeoutException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The LockTimeoutException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getMessage	Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

LockTimeoutException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Returns the detail message string of this exception. The returned String value includes the request queue details and the message that was provided to the constructor.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the detailed message string of this LockTimeoutException instance.

See Also

LockTimeoutException Class
LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

The LockTimeoutException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
LockRequestQueueDetails	Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred.
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

LockTimeoutException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets detailed information about the state of the lock on the lock request queue at the time the lock timeout occurred.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

Gets the value that was set in the LockRequestQueueDetails property. If the LockRequestQueueDetails property was not previously set for this exception, the return value is null.

Sets the details of the lock requests on the lock request queue at the time the lock timeout occurred.

See Also

LockTimeoutException Class
LockTimeoutException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A MultiplePartitionWriteException exception is a base exception for client/server operations when a user attempts to write to multiple remote partitions on remote servers in the same transaction.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map MultiplePartitionWriteException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

MultiplePartitionWriteException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MultiplePartitionWriteException type exposes the following members.

Constructors

Name	Description
MultiplePartitionWriteException	Initializes a new instance of the MultiplePartitionWriteException class.
MultiplePartitionWriteException(Exception)	Initializes a new instance of the MultiplePartitionWriteException class with a specified exception cause.
MultiplePartitionWriteException(String)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message.
MultiplePartitionWriteException(String, Exception)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

MultiplePartitionWriteException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
MultiplePartitionWriteException	Initializes a new instance of the MultiplePartitionWriteException class.
MultiplePartitionWriteException(Exception)	Initializes a new instance of the MultiplePartitionWriteException class with a specified exception cause.
MultiplePartitionWriteException(String)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message.
MultiplePartitionWriteException(String, Exception)	Initializes a new instance of the MultiplePartitionWriteException class with the specified error message and exception cause.

Back to Top

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the MultiplePartitionWriteException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

MultiplePartitionWriteException Class
MultiplePartitionWriteException Members
MultiplePartitionWriteException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MultiplePartitionWriteException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

MultiplePartitionWriteException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The MultiplePartitionWriteException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

MultiplePartitionWriteException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An OptimisticCollisionException occurs when an optimistic locking strategy is used and more than one update transaction collides on the same map entry of an ObjectGrid instance. The first transaction to commit updates the version object for the map entry. Other transactions that read this same map entry before committing have the previous version object. When the other transactions try to commit, the version object that is read does not match the version that was last committed. Therefore, other transactions are prevented from updating a map entry with stale data.

The default OptimisticCallback plug-in is used by the run time if an implementation is not provided by the application. If a well-constructed equals(Object) method is not on your value object, this exception occurs because the entire value object is used as the version object.

Because this exception indicates that the map entry contains stale data, stale map entries or entries as identified by the key parameter that is passed to the OptimisticCollisionException(String, String, String, Object) method are invalidated. If this exception is thrown by a Loader plug-in and a null reference is used as the key parameter by the loader, the run time assumes that the loader does not know which entry caused the exception. In this scenario, the LogSequence object is passed to the Loader.batchUpdate(TxID, LogSequence) method to determine which map entries to invalidate. Each LogElement entry in the LogSequence object that is type update or delete is invalidated.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map OptimisticCollisionException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OptimisticCollisionException type exposes the following members.

Constructors

Name	Description
OptimisticCollisionException	Initializes a new instance of the OptimisticCollisionException class with the specified error message, data grid name, map name, and keys that caused the exception.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
getGridName	Gets the name of the ObjectGrid instance in which the optimistic collision occurred.
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getKey	Gets key object or array of key objects that caused the OptimisticCollisionException to occur.
getMapName	Gets the map name in which the OptimisticCollisionException exception occurred.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
setKey	Set the key that caused this exception to occur.
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

OptimisticCollisionException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the OptimisticCollisionException class with the specified error message, data grid name, map name, and keys that caused the exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

gridName
Type: System String
Specfies the name of the data grid to which the map belongs.

mapName
Type: System String
Specifies the name of the map.

key
Type: System Object
Specifies the key or array of keys that caused the optimistic collision exception to occur.

Remarks

If more than a single key caused the exception, use an array object for this parameter. Each array element identifies a single map entry that caused the exception to occur. Using array elements is useful when a Loader uses the batch update support of a Java Database Connectivity (JDBC) driver. Pass a null reference if you are unable to determine which key or set of keys caused this exception to occur.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OptimisticCollisionException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
getGridName	Gets the name of the ObjectGrid instance in which the optimistic collision occurred.
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getKey	Gets key object or array of key objects that caused the OptimisticCollisionException to occur.
getMapName	Gets the map name in which the OptimisticCollisionException exception occurred.
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
setKey	Set the key that caused this exception to occur.
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

OptimisticCollisionException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the name of the ObjectGrid instance in which the optimistic collision occurred.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the name of the ObjectGrid instance.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets key object or array of key objects that caused the OptimisticCollisionException to occur.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the key object or array of key objects that caused the exception to occur.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets the map name in which the OptimisticCollisionException exception occurred.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Specifies the map name where the exception occurred.

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Set the key that caused this exception to occur.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

key
Type: System Object
the key or array of key objects that caused the exception

See Also

OptimisticCollisionException Class
OptimisticCollisionException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The OptimisticCollisionException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

OptimisticCollisionException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies one or more attributes to use to calculate the partition hash code.

The PartitionKey attribute can be specified for the class using a path syntax to identify a single attribute. Multiple attributes can be specified using additional PartitionKey annotations on each field, using the Order attribute to specify the order in which the hash codes will be calculated.

The PartitionKey attribute is not inheritable.

Inheritance Hierarchy

System Object
  System Attribute
    IBM.WebSphere.Caching.Map PartitionKeyAttribute

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

The following examples illustrate how to identify a top-level, primitive field:

Simple class attribute example:

Copy to ClipboardPrint

```
[PartitionKey("deptId")] class Employee { int empId; int deptId; }
```

Simple field attribute example:

Copy to ClipboardPrint

```
class Employee { int empId; [PartitionKey] int deptId; }
```

Simple, multiple field attribute example:

Copy to ClipboardPrint

```
class Employee { int empId; [PartitionKey(order=0)] int deptId; [PartitionKey(order=1)] String coun
```

The following examples illustrate how to address an attribute inside an embedded class. In this case, the path separater is defined in the data grid as a "." character (the default for eXtreme Data Format):

Embedded class annotation example:

Copy to ClipboardPrint

```
[PartitionKey("deptKey.id")] class Employee { DepartmentKey deptKey; } class DepartmentKey { int id
```

Embedded field annotation example:

Copy to ClipboardPrint

```
class Employee { [PartitionKey("id")] DepartmentKey deptKey; } class DepartmentKey { int id; }
```

See Also

PartitionKeyAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The PartitionKeyAttribute type exposes the following members.

Constructors

Name	Description
PartitionKeyAttribute	An attribute-less constructor for use when defined on a field, where the field name is the attribute.
PartitionKeyAttribute(String)	Specifies a single attribute to use to calculate the partition hash code.

Back to Top

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

Properties

Name	Description
AttributeName	Identifies the path to the attribute that should be included as part of the partition key.
Order	Order of the fields that contribute to the partition calculation. The order values must be unique for all PartitionKey attributes defined on a key class.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

PartitionKeyAttribute Class
IBM.WebSphere.Caching.Map Namespace

information see the WebSphere™ eXtreme Scale Library.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
PartitionKeyAttribute	An attribute-less constructor for use when defined on a field, where the field name is the attribute.
PartitionKeyAttribute(String)	Specifies a single attribute to use to calculate the partition hash code.

Back to Top

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An attribute-less constructor for use when defined on a field, where the field name is the attribute.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
PartitionKeyAttribute Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies a single attribute to use to calculate the partition hash code.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

attributeName
Type: System String
The attribute name.

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
PartitionKeyAttribute Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The PartitionKeyAttribute type exposes the following members.

Methods

Name	Description
Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
GetType	Gets the Type of the current instance. (Inherited from Object.)
IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Returns a String that represents the current Object. (Inherited from Object.)

Back to Top

See Also

PartitionKeyAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The PartitionKeyAttribute type exposes the following members.

Properties

Name	Description
AttributeName	Identifies the path to the attribute that should be included as part of the partition key.
Order	Order of the fields that contribute to the partition calculation. The order values must be unique for all PartitionKey attributes defined on a key class.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Back to Top

See Also

PartitionKeyAttribute Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Identifies the path to the attribute that should be included as part of the partition key.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

The attribute path.

See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Order of the fields that contribute to the partition calculation. The order values must be unique for all PartitionKey attributes defined on a key class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Field Value

The field order.
See Also

PartitionKeyAttribute Class
PartitionKeyAttribute Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

ReadOnlyException Class IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A ReadOnlyException exception occurs when a modify operation is attempted on a read-only map.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map ReadOnlyException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ReadOnlyException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReadOnlyException type exposes the following members.

Constructors

Name	Description
ReadOnlyException	Initializes a new instance of the ReadOnlyException> class.
ReadOnlyException(Exception)	Initializes a new instance of the ReadOnlyException class with a specified exception cause.
ReadOnlyException(String)	Initializes a new instance of the ReadOnlyException class with the specified error message.
ReadOnlyException(String, Exception)	Initializes a new instance of the ReadOnlyException class with the specified error message and exception cause.

Back to Top

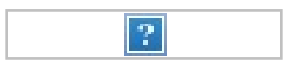
Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)

	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

ReadOnlyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ReadOnlyException	Initializes a new instance of the ReadOnlyException> class.
ReadOnlyException(Exception)	Initializes a new instance of the ReadOnlyException class with a specified exception cause.
ReadOnlyException(String)	Initializes a new instance of the ReadOnlyException class with the specified error message.
ReadOnlyException(String, Exception)	Initializes a new instance of the ReadOnlyException class with the specified error message and exception cause.

Back to Top

See Also

ReadOnlyException Class
ReadOnlyException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReadOnlyException> class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ReadOnlyException Class
ReadOnlyException Members
ReadOnlyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReadOnlyException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ReadOnlyException Class
ReadOnlyException Members
ReadOnlyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReadOnlyException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
: Specifies the error message that explains the reason for the exception.

See Also

ReadOnlyException Class
ReadOnlyException Members
ReadOnlyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ReadOnlyException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ReadOnlyException Class
ReadOnlyException Members
ReadOnlyException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReadOnlyException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ReadOnlyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ReadOnlyException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ReadOnlyException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TargetNotAvailableException occurs when a remote target was not found or was not reachable.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map TargetNotAvailableException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TargetNotAvailableException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TargetNotAvailableException type exposes the following members.

Constructors

Name	Description
TargetNotAvailableException(String)	Initializes a new instance of the TargetNotAvailableException class with the specified error message.
TargetNotAvailableException(String, Exception)	Initializes a new instance of the TargetNotAvailableException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Gets or sets the name of the application or the object that causes the

	Source	error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

TargetNotAvailableException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TargetNotAvailableException(String)	Initializes a new instance of the TargetNotAvailableException class with the specified error message.
TargetNotAvailableException(String, Exception)	Initializes a new instance of the TargetNotAvailableException class with the specified error message and exception cause.

Back to Top

See Also

TargetNotAvailableException Class
TargetNotAvailableException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TargetNotAvailableException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
The error message that explains the reason for the exception.

See Also

TargetNotAvailableException Class
TargetNotAvailableException Members
TargetNotAvailableException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TargetNotAvailableException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
: Type: System String
Specifies the error message that explains the reason for the exception.

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TargetNotAvailableException Class
TargetNotAvailableException Members
TargetNotAvailableException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TargetNotAvailableException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

TargetNotAvailableException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TargetNotAvailableException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TargetNotAvailableException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A TransactionAffinityException exception occurs during server failover for inflight transactions. Applications can try the transaction again.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LoaderException
        IBM.WebSphere.Caching.Map UnavailableServiceException
          IBM.WebSphere.Caching.Map TransactionAffinityException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

TransactionAffinityException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAffinityException type exposes the following members.

Constructors

Name	Description
TransactionAffinityException	Initializes a new instance of the TransactionAffinityException class.
TransactionAffinityException(Exception)	Initializes a new instance of the TransactionAffinityException class with a specified exception cause.
TransactionAffinityException(String)	Initializes a new instance of the TransactionAffinityException class with the specified error message.
TransactionAffinityException(String, Exception)	Initializes a new instance of the TransactionAffinityException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	ReplicationGroup	Gets or sets the replication group ID for this exception. (Inherited from UnavailableServiceException.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

TransactionAffinityException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
TransactionAffinityException	Initializes a new instance of the TransactionAffinityException class.
TransactionAffinityException(Exception)	Initializes a new instance of the TransactionAffinityException class with a specified exception cause.
TransactionAffinityException(String)	Initializes a new instance of the TransactionAffinityException class with the specified error message.
TransactionAffinityException(String, Exception)	Initializes a new instance of the TransactionAffinityException class with the specified error message and exception cause.

Back to Top

See Also

TransactionAffinityException Class
TransactionAffinityException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAffinityException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

TransactionAffinityException Class
TransactionAffinityException Members
TransactionAffinityException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAffinityException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionAffinityException Class
TransactionAffinityException Members
TransactionAffinityException Overload
IBM.WebSphere.Caching.Map Namespace

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAffinityException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

TransactionAffinityException Class
TransactionAffinityException Members
TransactionAffinityException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the TransactionAffinityException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

TransactionAffinityException Class
TransactionAffinityException Members
TransactionAffinityException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAffinityException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionAffinityException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The TransactionAffinityException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
ReplicationGroup	Gets or sets the replication group ID for this exception. (Inherited from UnavailableServiceException.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

TransactionAffinityException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Every grid map has an optional, built in timed based evictor that is referred to as the "time to live" evictor or TTL evictor. Each grid map entry has an expiration time that determines how long the entry is allowed to live in the grid map. When the expiration time is reached, the TTL evictor causes the expired entry to be evicted from the grid map. This enum defines the TTLType value constants that determine how the the expiration time is computed for a map entry.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Members

Member name	Value	Description
None	0	Indicates an entry has no expiration time and is allowed to live in the BackingMap until the application explicitly removes or invalidates the entry or a user defined evictor evicts it.
CreationTime	1	Indicates an entry expiration time is the sum of the creation time of the entry plus the "time to live" value.
LastAccessTime	2	Indicates an entry expiration time is the sum of the last access time of the entry, the time the entry was updated or read, plus the "time to live" value.
LastUpdateTime	3	Indicates an entry expiration time is the sum of the last update time of the entry plus the "time to live" value

See Also

IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An UnavailableServiceException exception occurs when all servers are not running, or when all services are not available even though servers are running.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map LoaderException
        IBM.WebSphere.Caching.Map UnavailableServiceException
          IBM.WebSphere.Caching.Map TransactionAffinityException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

UnavailableServiceException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The UnavailableServiceException type exposes the following members.

Constructors

Name	Description
UnavailableServiceException	Initializes a new instance of the UnavailableServiceException class.
UnavailableServiceException(Exception)	Initializes a new instance of the UnavailableServiceException class with a specified exception cause.
UnavailableServiceException(String)	Initializes a new instance of the UnavailableServiceException class with the specified error message.
UnavailableServiceException(String, Exception)	Initializes a new instance of the UnavailableServiceException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Replication Group	Gets or sets the replication group ID for this exception.
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

UnavailableServiceException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
UnavailableServiceException	Initializes a new instance of the UnavailableServiceException class.
UnavailableServiceException(Exception)	Initializes a new instance of the UnavailableServiceException class with a specified exception cause.
UnavailableServiceException(String)	Initializes a new instance of the UnavailableServiceException class with the specified error message.
UnavailableServiceException(String, Exception)	Initializes a new instance of the UnavailableServiceException class with the specified error message and exception cause.

Back to Top

See Also

UnavailableServiceException Class
UnavailableServiceException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UnavailableServiceException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

UnavailableServiceException Class
UnavailableServiceException Members
UnavailableServiceException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UnavailableServiceException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

UnavailableServiceException Class
UnavailableServiceException Members
UnavailableServiceException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UnavailableServiceException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

UnavailableServiceException Class
UnavailableServiceException Members
UnavailableServiceException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UnavailableServiceException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

UnavailableServiceException Class
UnavailableServiceException Members
UnavailableServiceException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The UnavailableServiceException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

UnavailableServiceException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

[Send Feedback]

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The UnavailableServiceException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
ReplicationGroup	Gets or sets the replication group ID for this exception.
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

UnavailableServiceException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Gets or sets the replication group ID for this exception.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Remarks

Gets the value that was set in the ReplicationGroup ID property, or null if the ReplicationGroup property was not previously set for this exception.

Sets the the ReplicationGroup ID for this exception.

See Also

UnavailableServiceException Class
UnavailableServiceException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An UndefinedMapException exception occurs to indicate that the map that an application tried to access is not defined in the ObjectGrid.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridException
      IBM.WebSphere.Caching.Map UndefinedMapException

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

UndefinedMapException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The UndefinedMapException type exposes the following members.

Constructors

	Name	Description
	UndefinedMapException	Initializes a new instance of the UndefinedMapException class.
	UndefinedMapException(Exception)	Initializes a new instance of the UndefinedMapException class with a specified exception cause.
	UndefinedMapException(String)	Initializes a new instance of the UndefinedMapException class with the specified error message.
	UndefinedMapException(String, Exception)	Initializes a new instance of the UndefinedMapException class with the specified error message and exception cause.

Back to Top

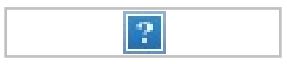
Methods

	Name	Description
	Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
	Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
	GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
	GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

	Name	Description
	Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception.

		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

UndefinedMapException Class

IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
UndefinedMapException	Initializes a new instance of the UndefinedMapException class.
UndefinedMapException(Exception)	Initializes a new instance of the UndefinedMapException class with a specified exception cause.
UndefinedMapException(String)	Initializes a new instance of the UndefinedMapException class with the specified error message.
UndefinedMapException(String, Exception)	Initializes a new instance of the UndefinedMapException class with the specified error message and exception cause.

Back to Top

See Also

UndefinedMapException Class
UndefinedMapException Members
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UndefinedMapException class.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

UndefinedMapException Class
UndefinedMapException Members
UndefinedMapException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UndefinedMapException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

UndefinedMapException Class
UndefinedMapException Members
UndefinedMapException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UndefinedMapException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

UndefinedMapException Class
UndefinedMapException Members
UndefinedMapException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the UndefinedMapException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Map
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

UndefinedMapException Class
UndefinedMapException Members
UndefinedMapException Overload
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The UndefinedMapException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

UndefinedMapException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The UndefinedMapException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

UndefinedMapException Class
IBM.WebSphere.Caching.Map Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The IBM.WebSphere.Caching.Security namespace includes the application programming interfaces specific to security.

The eXtreme Scale client supports transport security and authenticiation and authorization using the ICredentialGenerator interface. Security is configured using a client properties file.

Classes

Class	Description
AccessControlException	This Java exception is thrown when an access control check indicates that access should not be granted.
CannotGenerateCredentialException	A CannotGenerateCredentialException exception occurs if a credential cannot be generated.
ExpiredCredentialException	An ExpiredCredentialException exception indicates that the credential that was used for authentication is expired.
GridSecurityException	An GridSecurityException exception occurs for general security exceptions.

Interfaces

Interface	Description
ICredential	Specifies the interface definition of the Credential class. Application developers implement this interface to create a custom ICredential implementation. An instance is created with a custom ICredentialGenerator.
ICredentialGenerator	Specifies the interface definition of the CredentialGenerator class. Application developers implement this interface to generate custom ICredential objects to use for authentication and authorization.

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

This Java exception is thrown when an access control check indicates that access should not be granted.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching GridServerRuntimeException
      IBM.WebSphere.Caching.Security AccessControlException

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

AccessControlException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AccessControlException type exposes the following members.

Constructors

Name	Description
AccessControlException	Initializes a new instance of the AccessControlException class.
AccessControlException(Exception)	Initializes a new instance of the AccessControlException class with a specified exception cause.
AccessControlException(String)	Initializes a new instance of the AccessControlException class with the specified error message.
AccessControlException(String, Exception)	Initializes a new instance of the AccessControlException> class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Provides a user-readable representation of this GridServerRuntimeException exception. (Inherited from GridServerRuntimeException.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)

	JavaExceptionClassName	Identifies the Java Exception class name that this exception identifies. (Inherited from GridServerRuntimeException.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

AccessControlException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
AccessControlException	Initializes a new instance of the AccessControlException class.
AccessControlException(Exception)	Initializes a new instance of the AccessControlException class with a specified exception cause.
AccessControlException(String)	Initializes a new instance of the AccessControlException class with the specified error message.
AccessControlException(String, Exception)	Initializes a new instance of the AccessControlException> class with the specified error message and exception cause.

Back to Top

See Also

AccessControlException Class
AccessControlException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AccessControlException class.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

AccessControlException Class
AccessControlException Members
AccessControlException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AccessControlException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
  Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

AccessControlException Class
AccessControlException Members
AccessControlException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AccessControlException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

AccessControlException Class
AccessControlException Members
AccessControlException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the AccessControlException> class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

innerException
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

AccessControlException Class
AccessControlException Members
AccessControlException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AccessControlException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Provides a user-readable representation of this GridServerRuntimeException exception. (Inherited from GridServerRuntimeException.)

Back to Top

See Also

AccessControlException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The AccessControlException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
JavaExceptionClassName	Identifies the Java Exception class name that this exception identifies. (Inherited from GridServerRuntimeException.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

AccessControlException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

A CannotGenerateCredentialException exception occurs if a credential cannot be generated.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching.Security GridSecurityException
      IBM.WebSphere.Caching.Security CannotGenerateCredentialException

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

CannotGenerateCredentialException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CannotGenerateCredentialException type exposes the following members.

Constructors

Name	Description
CannotGenerateCredentialException	Initializes a new instance of the CannotGenerateCredentialException class.
CannotGenerateCredentialException(Exception)	Initializes a new instance of the CannotGenerateCredentialException class with the specified exception cause.
CannotGenerateCredentialException(String)	Initializes a new instance of the CannotGenerateCredentialException class with the specified error message.
CannotGenerateCredentialException(String, Exception)	Initializes a new instance of the CannotGenerateCredentialException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)

	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top
See Also

CannotGenerateCredentialException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
CannotGenerateCredentialException	Initializes a new instance of the CannotGenerateCredentialException class.
CannotGenerateCredentialException(Exception)	Initializes a new instance of the CannotGenerateCredentialException class with the specified exception cause.
CannotGenerateCredentialException(String)	Initializes a new instance of the CannotGenerateCredentialException class with the specified error message.
CannotGenerateCredentialException(String, Exception)	Initializes a new instance of the CannotGenerateCredentialException class with the specified error message and exception cause.

Back to Top

See Also

CannotGenerateCredentialException Class
CannotGenerateCredentialException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CannotGenerateCredentialException class.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

CannotGenerateCredentialException Class
CannotGenerateCredentialException Members
CannotGenerateCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CannotGenerateCredentialException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

CannotGenerateCredentialException Class
CannotGenerateCredentialException Members
CannotGenerateCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CannotGenerateCredentialException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

CannotGenerateCredentialException Class
CannotGenerateCredentialException Members
CannotGenerateCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the CannotGenerateCredentialException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

CannotGenerateCredentialException Class
CannotGenerateCredentialException Members
CannotGenerateCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CannotGenerateCredentialException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

CannotGenerateCredentialException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The CannotGenerateCredentialException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

CannotGenerateCredentialException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An ExpiredCredentialException exception indicates that the credential that was used for authentication is expired.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching.Security GridSecurityException
      IBM.WebSphere.Caching.Security ExpiredCredentialException

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ExpiredCredentialException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ExpiredCredentialException type exposes the following members.

Constructors

Name	Description
ExpiredCredentialException	Initializes a new instance of the ExpiredCredentialException class.
ExpiredCredentialException(Exception)	Initializes a new instance of the ExpiredCredentialException class with the specified exception cause.
ExpiredCredentialException(String)	Initializes a new instance of the ExpiredCredentialException class with the specified error message.
ExpiredCredentialException(String, Exception)	Initializes a new instance of the ExpiredCredentialException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)

	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ExpiredCredentialException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
ExpiredCredentialException	Initializes a new instance of the ExpiredCredentialException class.
ExpiredCredentialException(Exception)	Initializes a new instance of the ExpiredCredentialException class with the specified exception cause.
ExpiredCredentialException(String)	Initializes a new instance of the ExpiredCredentialException class with the specified error message.
ExpiredCredentialException(String, Exception)	Initializes a new instance of the ExpiredCredentialException class with the specified error message and exception cause.

Back to Top

See Also

ExpiredCredentialException Class
ExpiredCredentialException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ExpiredCredentialException class.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

Copy to ClipboardPrint

See Also

ExpiredCredentialException Class
ExpiredCredentialException Members
ExpiredCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ExpiredCredentialException class with the specified exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause of the exception is nonexistent or unknown.

See Also

ExpiredCredentialException Class
ExpiredCredentialException Members
ExpiredCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ExpiredCredentialException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

ExpiredCredentialException Class
ExpiredCredentialException Members
ExpiredCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the ExpiredCredentialException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

ExpiredCredentialException Class
ExpiredCredentialException Members
ExpiredCredentialException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ExpiredCredentialException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

ExpiredCredentialException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ExpiredCredentialException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

ExpiredCredentialException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

An GridSecurityException exception occurs for general security exceptions.

Inheritance Hierarchy

System Object
  System Exception
    IBM.WebSphere.Caching.Security GridSecurityException
      IBM.WebSphere.Caching.Security CannotGenerateCredentialException
      IBM.WebSphere.Caching.Security ExpiredCredentialException

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

GridSecurityException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridSecurityException type exposes the following members.

Constructors

Name	Description
GridSecurityException	Initializes a new instance of the GridSecurityException class.
GridSecurityException(Exception)	Initializes a new instance of the GridSecurityException class with a specified exception cause.
GridSecurityException(String)	Initializes a new instance of the GridSecurityException class with the specified error message.
GridSecurityException(String, Exception)	Initializes a new instance of the GridSecurityException class with the specified error message and exception cause.

Back to Top

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception.

		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridSecurityException Class

IBM.WebSphere.Caching.Security Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Overload List

Name	Description
GridSecurityException	Initializes a new instance of the GridSecurityException class.
GridSecurityException(Exception)	Initializes a new instance of the GridSecurityException class with a specified exception cause.
GridSecurityException(String)	Initializes a new instance of the GridSecurityException class with the specified error message.
GridSecurityException(String, Exception)	Initializes a new instance of the GridSecurityException class with the specified error message and exception cause.

Back to Top

See Also

GridSecurityException Class
GridSecurityException Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridSecurityException class.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript

See Also

GridSecurityException Class
GridSecurityException Members
GridSecurityException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridSecurityException class with a specified exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

cause
: Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridSecurityException Class
GridSecurityException Members
GridSecurityException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridSecurityException class with the specified error message.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

See Also

GridSecurityException Class
GridSecurityException Members
GridSecurityException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Initializes a new instance of the GridSecurityException class with the specified error message and exception cause.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

message
Type: System String
Specifies the error message that explains the reason for the exception.

cause
Type: System Exception
Specifies the exception that is the cause of the current exception. A null value is permitted and indicates that the cause is nonexistent or unknown.

See Also

GridSecurityException Class
GridSecurityException Members
GridSecurityException Overload
IBM.WebSphere.Caching.Security Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridSecurityException type exposes the following members.

Methods

Name	Description
Equals	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Back to Top

See Also

GridSecurityException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The GridSecurityException type exposes the following members.

Properties

Name	Description
Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Back to Top

See Also

GridSecurityException Class
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

ICredential Interface IBM WebSphere™ eXtreme Scale Client for .NET API Specification
Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the interface definition of the Credential class.

Application developers implement this interface to create a custom ICredential implementation. An instance is created with a custom ICredentialGenerator.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICredential Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

ICredential Members IBM WebSphere™ eXtreme Scale Client for .NET API Specification

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICredential type exposes the following members.

Methods

Name	Description
Equals	Checks if two credential objects are equal. Two credential objects are considered equal if they represent the same identity and security information.
GetHashCode	Specifies a hash code representation of the credential object.

Back to Top

See Also

ICredential Interface
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback
IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICredential type exposes the following members.
Methods

	Name	Description
	Equals	Checks if two credential objects are equal. Two credential objects are considered equal if they represent the same identity and security information.
	GetHashCode	Specifies a hash code representation of the credential object.

Back to Top
See Also

ICredential Interface
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Checks if two credential objects are equal. Two credential objects are considered equal if they represent the same identity and security information.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Parameters

credential
Type: IBM.WebSphere.Caching.Security ICredential
Specifies the object that is being tested for equality with the selected object.

Return Value

Returns true if both credential objects are equivalent.
See Also

ICredential Interface
ICredential Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies a hash code representation of the credential object.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

Return Value

Returns the hash code representation of the credential object.

See Also

ICredential Interface
ICredential Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the interface definition of the CredentialGenerator class.

Application developers implement this interface to generate custom ICredential objects to use for authentication and authorization.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0
Syntax

VB
C#
C++
F#
JScript

See Also

ICredentialGenerator Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICredentialGenerator type exposes the following members.

Properties

Name	Description
Credential	Specifies the Credential that represents the client. A CannotGenerateCredentialException exception occurs if generation of the credential for the client fails.
Properties	Specifies the user-defined properties for the CredentialGenerator factory. These properties are used to influence the contents of the credential that is generated by the CredentialGenerator factory. This property value is set to the credentialGeneratorProps property value in the client security configuration property file.

Back to Top

See Also

ICredentialGenerator Interface
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

The ICredentialGenerator type exposes the following members.

Properties

Name	Description
Credential	Specifies the Credential that represents the client. A CannotGenerateCredentialException exception occurs if generation of the credential for the client fails.
Properties	Specifies the user-defined properties for the CredentialGenerator factory. These properties are used to influence the contents of the credential that is generated by the CredentialGenerator factory. This property value is set to the credentialGeneratorProps property value in the client security configuration property file.

Back to Top

See Also

ICredentialGenerator Interface
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

Send Feedback

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the Credential that represents the client. A CannotGenerateCredentialException exception occurs if generation of the credential for the client fails.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICredentialGenerator Interface
ICredentialGenerator Members
IBM.WebSphere.Caching.Security Namespace

Send Feedback on this topic to WAS Documentation Team.

IBM WebSphere™ eXtreme Scale Client for .NET Release 8.6.0.0 API Specification

Specifies the user-defined properties for the CredentialGenerator factory.

These properties are used to influence the contents of the credential that is generated by the CredentialGenerator factory. This property value is set to the credentialGeneratorProps property value in the client security configuration property file.

Namespace: IBM.WebSphere.Caching.Security
Assembly: Client.Api (in Client.Api.dll) Version: 8.6.0.0

Syntax

VB
C#
C++
F#
JScript
Copy to ClipboardPrint

See Also

ICredentialGenerator Interface
ICredentialGenerator Members
IBM.WebSphere.Caching.Security Namespace

Paramètres de l'interface utilisateur

Ces informations de référence décrivent les paramètres que vous pouvez afficher et configurer dans les pages de la console d'administration WebSphere Application Server et ailleurs.

Pour ouvrir la table des matières dans l'emplacement de ces informations de référence, cliquez sur le bouton **Afficher dans la table des matières** () sur la bordure du centre de documentation. Par exemple, si vous avez trouvé cette page en effectuant une recherche dans le centre de documentation ou à partir d'un moteur de recherche Internet, tel que Google, cliquez sur le bouton pour que le centre de documentation vous indique l'emplacement de cette page dans l'arborescence de navigation du centre de documentation Vous serez en mesure de parcourir le contenu sous cette entrée de navigation.

Paramètres de gestion des sessions eXtreme Scale
Vous pouvez configurer les applications WebSphere Application Server pour utiliser WebSphere eXtreme Scale ou un WebSphere DataPower XC10 Appliance pour la persistance de session.

Collection de domaines de service de catalogue
Cette page permet de gérer les domaines de service de catalogue. Les domaines de service de catalogue définissent un groupe de serveurs de catalogue qui gèrent le positionnement de fragments et surveille l'état des serveurs de conteneur dans la grille de données.

Paramètres du domaine de service de catalogue
Cette page permet de gérer les paramètres d'un domaine de service de catalogue spécifique. Les domaines de service de catalogue définissent un groupe de serveurs de catalogue qui gèrent le positionnement de fragments et surveille l'état des serveurs de conteneur dans la grille de données. Vous pouvez définir un domaine de service de catalogue figurant dans la même cellule que le gestionnaire de déploiement. Vous pouvez également définir des domaines de service de catalogue distants si votre configuration WebSphere eXtreme Scale se trouve dans une cellule différente ou que votre grille de données est composée de processus Java™ SE.

Propriétés de sécurité du client
Utilisez cette page pour définir la sécurité client d'un domaine de service de catalogue. Ces paramètres s'appliquent à tous les serveurs dans votre domaine de service de catalogue. Ces propriétés peuvent être remplacées en définissant un fichier `splicer.properties` avec la propriété personnalisée com.ibm.websphere.xs.sessionFilterProps ou en raccordant le fichier EAR d'application.

Propriétés personnalisées du domaine de service de catalogue
Vous pouvez modifier davantage la configuration du domaine de service de catalogue en définissant des propriétés personnalisées.

Rubrique parent : Référence

Paramètres de gestion des sessions eXtreme Scale

Vous pouvez configurer les applications WebSphere Application Server pour utiliser WebSphere eXtreme Scale ou un WebSphere DataPower XC10 Appliance pour la persistance de session.

Vous pouvez modifier ces paramètres dans l'assistant d'installation des applications d'entreprise ou dans les pages d'information de l'application ou du serveur :

- Version 7.0 : **Applications** > **Nouvelle application** > **Nouvelle applications d'enterprise** et choisissez le chemin détaillé pour créer l'application.
- Version 7.0 : **Applications** > **Types d'application** > **Applications d'entreprise WebSphere** > ***application_name*** > **Propriétés du module Web** > **Gestion de sessionmanagement** > **Paramètres de gestion de session**
- Version 7.0 : **Serveurs** > **Types de serveur** > **Serveurs d'applications WebSphere** > ***server_name*** > **Paramètres de conteneur** > **Paramètres de gestion de session**

Rubrique parent : Paramètres de l'interface utilisateur

Activer la gestion des sessions

Permet à la gestion des sessions d'utiliser WebSphere eXtreme Scale imbriquée ou une grille de données distante ou un WebSphere DataPower XC10 Appliance pour la persistance de session.

Gestion de la persistance des sessions par

Indique comment la persistance des sessions est gérée. Vous pouvez sélectionner l'une des options suivantes :

- WebSphere DataPower XC10 Appliance
- Grille de données distante eXtreme Scale
- Grille de données imbriquée eXtreme Scale

Les autres paramètres que vous configurez varient en fonction du mécanisme de persistance des sessions sélectionné.

Paramètres WebSphere DataPower XC10 Appliance

Les paramètres suivants sont spécifiques de la configuration de WebSphere DataPower XC10 Appliance pour la persistance de session.

Adresse IP ou nom d'hôte de WebSphere DataPower XC10 Appliance

Indique l'adresse IP ou le nom d'hôte de l'appliance à utiliser pour stocker les données des sessions.

Information d'identification administrative IBM® WebSphere DataPower XC10 Appliance

Indiquez le **nom d'utilisateur** et le **mot de passe** que vous utilisez pour vous connecter à l'interface utilisateur de DataPower XC10 Appliance. Cliquez sur **Tester la connexion...** pour tester la connexion à l'appliance.

Préférences de persistance des sessions

Indique la grille de données dans laquelle les sessions sont conservées. Vous pouvez sélectionner l'une des options suivantes :

- **Conserver les sessions dans une nouvelle grille de données sur IBM WebSphere DataPower XC10 Appliance**. Vous pouvez ensuite indiquer un **nom de grille de données**.
- **Conserver les sessions dans une grille de données existante sur IBM WebSphere DataPower XC10 Appliance**. Vous pouvez ensuite entrer ou rechercher un **nom de grille de données existant**.

Configuration d'une grille de données distante eXtreme Scale

Les paramètres suivants s'appliquent à la configuration de la grille distante eXtreme Scale pour la persistance des sessions :

Domaine de service de catalogue gérant la grille de données de session distante

Indique le domaine de service de catalogue à utiliser pour gérer les sessions.

Si aucun domaine de service de catalogue n'est affiché ou que vous souhaitez créer un domaine de service de catalogue nouveau, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **domaine de service de catalogue**.

Grille de données distante utilisée pour stocker les informations de session

Indique le nom de la grille de données du domaine de service de catalogue où les informations de session doivent être stockées. La liste des grilles distantes actives est renseignée lorsque vous sélectionnez un service de catalogue. La grille de données distante doit déjà être définie dans la configuration eXtreme Scale.

Configuration d'une grille de données imbriquée eXtreme Scale

Les paramètres suivants s'appliquent à une configuration eXtreme Scale imbriquée. Dans le scénario de configuration imbriquée eXtreme Scale, les processus eXtreme Scale sont hébergés par des processus WebSphere Application Server.

Configuration d'une grille de données imbriquée eXtreme Scale

- **Utiliser la configuration ObjectGrid par défaut**
- **Indiquer des fichiers de configuration ObjectGrid personnalisés**

  Chemin complet du fichier `objectgrid.xml` à copier dans la configuration

  Indique le chemin complet du fichier `objectgrid.xml` correspondant à la configuration à utiliser.

  Chemin complet du fichier `objectgriddeployment.xml` à copier dans la configuration

  Indique le chemin complet du fichier `objectgriddeployment.xml` correspondant à la configuration à utiliser.

Collection de domaines de service de catalogue

Cette page permet de gérer les domaines de service de catalogue. Les domaines de service de catalogue définissent un groupe de serveurs de catalogue qui gèrent le positionnement de fragments et surveille l'état des serveurs de conteneur dans la grille de données.

Pour afficher cette page de la console d'administration, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaine de service de catalogue**. Pour créer un domaine de service de catalogue, cliquez sur **Nouveau**. Pour supprimer un domaine de service de catalogue, sélectionnez le domaine à supprimer et cliquez sur **Supprimer**.

Rubrique parent : Paramètres de l'interface utilisateur

Tester la connexion

Lorsque vous cliquez sur le bouton de **test de la connexion**, tous les noeuds finals du domaine de service de catalogue défini sont interrogés un par un. Si l'un d'entre eux est disponible, il renvoie un message qui indique que la connexion au domaine de service de catalogue a abouti. Vous pouvez utiliser ce bouton pour tester la configuration des informations de connexion et de sécurité.

Définir la valeur par défaut

Indique le domaine de service de catalogue utilisé comme valeur par défaut. Sélectionnez un domaine de service de catalogue comme valeur par défaut et cliquez sur **Définir la valeur par défaut**. Un seul domaine de service de catalogue peut être sélectionné comme valeur par défaut.

Nom

Indique le nom du domaine de service de catalogue.

Valeur par défaut

Indique le domaine de service de catalogue de la liste qui est la valeur par défaut. Le domaine de service de catalogue par défaut est indiqué par l'icône :  .

Paramètres du domaine de service de catalogue

Cette page permet de gérer les paramètres d'un domaine de service de catalogue spécifique. Les domaines de service de catalogue définissent un groupe de serveurs de catalogue qui gèrent le positionnement de fragments et surveille l'état des serveurs de conteneur dans la grille de données. Vous pouvez définir un domaine de service de catalogue figurant dans la même cellule que le gestionnaire de déploiement. Vous pouvez également définir des domaines de service de catalogue distants si votre configuration WebSphere eXtreme Scale se trouve dans une cellule différente ou que votre grille de données est composée de processus Java™ SE.

Pour afficher cette page de la console d'administration, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaines de service de catalogue** > ***nom_domaine_service_catalogue***.

Rubrique parent : Paramètres de l'interface utilisateur

Tester la connexion

Lorsque vous cliquez sur le bouton de **test de la connexion**, tous les noeuds finals du domaine de service de catalogue défini sont interrogés un par un. Si l'un d'entre eux est disponible, il retourne un message qui indique que la connexion au domaine de service de catalogue a abouti. Vous pouvez utiliser ce bouton pour tester la configuration des informations de connexion et de sécurité.

Nom

Indique le nom du domaine de service de catalogue.

Activer ce domaine de service de catalogue en tant que valeur par défaut sauf si un autre domaine de service de catalogue est explicitement indiqué

Si vous cochez cette case, le domaine de service de catalogue sélectionné devient le domaine de service de catalogue par défaut de la cellule. Chaque profil de serveur dans la cellule qui est étendu avec le profil WebSphere eXtreme Scale appartient au domaine du service de catalogue sélectionné.

Si vous changez le domaine par défaut pour pointer vers un groupe de serveurs de catalogue différents, tous les conteneurs et clients font référence au nouveau domaine après leur redémarrage.

Activer les communications IBM eXtremeIO (XIO)

Indique si le domaine de service de catalogue utilise les communications XIO. Si vous ne sélectionnez pas cette option, ORB (Object Request Broker) est utilisé.

Remarque : Vous ne pouvez pas activer les communications XIO sur les serveurs distants depuis la console d'administration WebSphere Application Server. Activez XIO sur les serveurs distants lorsque vous démarrez les serveurs avec le script **`startXsServer`**.

Serveurs de catalogue

Indique une liste de serveurs de catalogue appartenant à ce domaine de service de catalogue.

Cliquez sur **Nouveau** pour ajouter un serveur de catalogue à la liste. Ce serveur de catalogue doit déjà être défini dans la configuration eXtreme Scale. Vous pouvez également modifier ou supprimer un serveur dans la liste en sélectionnant le noeud final, puis en cliquant sur **Editer** ou **Supprimer**. Définissez les propriétés suivantes pour chaque noeud final du serveur de catalogues :

Noeud final du serveur de catalogues

Indique le nom du serveur d'applications ou du serveur distant existant sur lequel le service de catalogues s'exécute. Un domaine de service de catalogue ne doit pas contenir une combinaison de serveurs d'applications et de serveurs distants existants.

- **Serveur d'applications existant** : définir le chemin d'un serveur d'applications, d'un agent de noeud ou de gestionnaire de déploiement dans la cellule. Un service de catalogue démarre automatiquement dans le serveur sélectionné. Sélectionnez un serveur dans la liste des serveurs d'applications existants. Tous les serveurs d'applications que vous définissez dans le domaine de service de catalogue doivent se trouver dans le même groupe central.
- **Serveur distant** : définit le nom d'hôte du serveur de catalogue distant.

  Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance : définit le nom du dispositif.

  Remarque : Vous ne pouvez pas activer les communications XIO sur les serveurs distants depuis la

console d'administration WebSphere Application Server. Activez XIO sur les serveurs distants lorsque vous démarrez les serveurs avec le script **startXsServer**.

Port du client

Spécifie le port utilisé pour la communication entre les serveurs de catalogues dans le domaine de service de catalogue. Cette valeur est nécessaire pour les serveurs de catalogue qui s'exécutent dans des processus WebSphere Application Server. Vous pouvez définir n'importe quel port comme valeur s'il n'est pas utilisé par un autre processus.

Port d'écoute

Indique le port utilisé pour établir des communications avec les clients. Cette valeur est obligatoire pour les noeuds finaux distants et elle doit correspondre à la valeur utilisée au démarrage du service de catalogue. Le port d'écoute est utilisé par les clients et les conteneurs pour communiquer avec le service de catalogue.
Pour les noeuds finaux distants WebSphere DataPower XC10 Appliance : utilisez la valeur 2809 pour les noeuds finaux distants du dispositif.

Statut

Tableau 1. Etat de noeud final de serveur de catalogues

Icône	Définition
	Inconnu
	Démarré
	Arrêté

Propriétés de sécurité du client

Utilisez cette page pour définir la sécurité client d'un domaine de service de catalogue. Ces paramètres s'appliquent à tous les serveurs dans votre domaine de service de catalogue. Ces propriétés peuvent être remplacées en définissant un fichier `splicer.properties` avec la propriété personnalisée com.ibm.websphere.xs.sessionFilterProps ou en raccordant le fichier EAR d'application.

Pour afficher cette page de la console d'administration, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **domaine de service de catalogue** > ***catalog_service_domain_name*** > **Propriétés de sécurité du client**.

Rubrique parent : Paramètres de l'interface utilisateur

Activer la sécurité du client

Spécifie que la sécurité du client est activée pour le serveur de catalogue. Le fichier des propriétés du serveur qui est associé au serveur de catalogue sélectionné doit avoir un paramètre **securityEnabled** correspondant dans le fichier des propriétés du serveur. Si ces paramètres ne correspondent pas, une exception est générée.

Authentification des données d'identification

Indique si l'authentification des données d'identification est imposée ou prise en charge.

Jamais

Aucune authentification n'est appliquée.

Requise

L'authentification des données d'identification est toujours appliquée. Si le serveur ne prend pas en charge l'authentification des données d'identification, le client ne peut pas se connecter au serveur.

Pris en charge

L'authentification des données d'identification est appliquée uniquement si le client et le serveur la prennent en charge.

Nombre de tentatives d'authentification

Spécifie le nombre de tentatives d'authentification si les données d'identification sont arrivées à expiration.

Si vous ne voulez pas réessayer l'authentification, définissez la valeur à `0`.

Classe de générateur de données d'identification

Indique la classe d'implémentation com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator pour que le client puisse extraire les données d'identification depuis l'objet CredentialGenerator.

Vous pouvez choisir depuis deux classes de générateurs d'identification de données prédéfinis ou vous pouvez définir un générateur personnalisé. Si vous choisissez un générateur personnalisé, vous devez indiquer le nom de la classe du générateur de données d'identification.

- com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator
- com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator
- Générateur de données d'identification personnalisé

Type de sujet

Indique si vous utilisez l'appelant J2EE ou le type de sujet J2EE runAs. Vous devez définir cette valeur lorsque vous choisissez le générateur de données d'identification WSTokenCredentialGenerator.

- **runAs** : le sujet contient le principal de l'exécution J2EE comme identité et l'exécution de J2EE en tant que données d'identification.
- **caller** : le sujet contient le principal de l'appelant J2EE et ses données d'identification.

ID utilisateur

Définissez un ID utilisateur lorsque vous utilisez l'implémentation de générateur de donnée d'identification UserPasswordCredentialGenerator.

Mot de passe

Définissez un mot de passe lorsque vous utilisez l'implémentation de générateur de donnée d'identification

UserPasswordCredentialGenerator.

Propriétés du générateur de données d'identification

Définissez les propriétés d'une classe d'implémentation CredentialGenerator personnalisée. Les propriétés sont définies dans l'objet avec la méthode setProperties(String). La valeur des propriétés du générateur de données d'identification est utilisée seulement si une valeur est spécifiée pour la zone **Classe du générateur de données d'identification**.

Propriétés personnalisées du domaine de service de catalogue

Vous pouvez modifier davantage la configuration du domaine de service de catalogue en définissant des propriétés personnalisées.

Pour afficher cette page de la console d'administration, cliquez sur **Administration du système** > **WebSphere eXtreme Scale** > **Domaine de service de catalogue** > **Propriétés personnalisées**. Pour créer une propriété personnalisée, cliquez sur **Nouveau**.

Rubrique parent : Paramètres de l'interface utilisateur

Nom

Indique le nom de la propriété personnalisée du domaine de service de catalogue.

Valeur

Indique une valeur pour la propriété personnalisée du domaine de service de catalogue.