9.0

IBM Message Service Client for .NET

Remarque

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 253.

Cette édition s'applique à la version 9 édition 0 d' IBM® MQ et à toutes les éditions et modifications ultérieures, sauf indication contraire dans les nouvelles éditions.

Table des matières

Présentation de IBM Message Service Client for .NET

IBM Message Service Client for .NET fournit une interface de programme d'application appelée XMS qui dispose du même ensemble d'interfaces que l'API Java Message Service (JMS). IBM Message Service Client for .NET comprend une implémentation complète de XMS, qui peut être utilisée par tout langage compatible avec .NET.

XMSprend en charge :

- une messagerie point-à-point
- une messagerie Publier/S'abonner
- Distribution synchrone des messages
- une distribution asynchrone de messages

Une application XMS peut échanger des messages avec les types suivants d'application :

- une application XMS
- une application IBM MQ classes for JMS
- une application IBM MQ native
- une application JMS qui utilise le fournisseur de messagerie IBM MQ par défaut

Une application XMS peut se connecter et utiliser les ressources des serveurs de messagerie suivants :

Gestionnaire de files d'attente IBM MQ
L'application peut se connecter en mode liaisons ou client.

WebSphere Application Server Service Integration Bus
L'application peut utiliser une connexion TCP/IP directe ou HTTP via TCP/IP.

IBM Integration Bus
Les messages sont transportés entre l'application et le courtier via WebSphere MQ Real-Time Transport. Les messages peuvent être distribués à l'application à l'aide de WebSphere MQ Multicast Transport.

En se connectant à un gestionnaire de files d'attente IBM MQ , une application XMS peut utiliser WebSphere MQ Enterprise Transport pour communiquer avec IBM Integration Bus. Une application XMS peut également effectuer des publications et s'abonner en se connectant à IBM MQ.

Concepts associés

«Styles de messagerie», à la page 6

«Modèle d'objet XMS», à la page 7
L'API XMS est une interface orientée objet. Le modèle d'objet XMS s'appuie sur le modèle d'objet JMS 1.1.

«Modèle de message XMS», à la page 10
Le modèle de message XMS est identique au modèle de message IBM MQ classes for JMS.

Présentation de IBM Message Service Client for .NET

IBM Message Service Client for .NET fournit une interface de programme d'application appelée XMS qui dispose du même ensemble d'interfaces que l'API Java Message Service (JMS). IBM Message Service Client for .NET comprend une implémentation complète de XMS, qui peut être utilisée par tout langage compatible avec .NET.

XMSprend en charge :

- une messagerie point-à-point
- une messagerie Publier/S'abonner
- Distribution synchrone des messages
- une distribution asynchrone de messages

Une application XMS peut échanger des messages avec les types suivants d'application :

- une application XMS
- une application IBM MQ classes for JMS
- une application IBM MQ native
- une application JMS qui utilise le fournisseur de messagerie IBM MQ par défaut

Une application XMS peut se connecter et utiliser les ressources des serveurs de messagerie suivants :

Gestionnaire de files d'attente IBM MQ
L'application peut se connecter en mode liaisons ou client.

WebSphere Application Server Service Integration Bus
L'application peut utiliser une connexion TCP/IP directe ou HTTP via TCP/IP.

IBM Integration Bus
Les messages sont transportés entre l'application et le courtier via WebSphere MQ Real-Time Transport. Les messages peuvent être distribués à l'application à l'aide de WebSphere MQ Multicast Transport.

En se connectant à un gestionnaire de files d'attente IBM MQ , une application XMS peut utiliser WebSphere MQ Enterprise Transport pour communiquer avec IBM Integration Bus. Une application XMS peut également effectuer des publications et s'abonner en se connectant à IBM MQ.

Concepts associés

«Styles de messagerie», à la page 6

«Modèle d'objet XMS», à la page 7
L'API XMS est une interface orientée objet. Le modèle d'objet XMS s'appuie sur le modèle d'objet JMS 1.1.

«Modèle de message XMS», à la page 10
Le modèle de message XMS est identique au modèle de message IBM MQ classes for JMS.

Styles de messagerie

XMS prend en charge les styles de messagerie point-à-point et Publier/S'abonner.

Les styles de messagerie sont également appelés domaines de messagerie.

une messagerie point-à-point

La forme commune de messagerie point-à-point utilise la mise en file d'attente. Dans le cas le plus simple, une application envoie un message à une autre application en identifiant, de manière implicite ou explicite, une file d'attente de destination. Le système de messagerie et de mise en file d'attente sous-jacent reçoit le message de l'application émettrice et l'achemine vers la file d'attente de destination. L'application de réception peut alors extraire le message de la file d'attente.

Si le système de messagerie et de mise en file d'attente sous-jacent contient IBM Integration Bus, IBM Integration Bus peut répliquer un message et acheminer des copies du message vers différentes files d'attente. Plusieurs applications peuvent donc recevoir le message. IBM Integration Bus peut également transformer un message et y ajouter des données.

Une caractéristique essentielle de la messagerie point-à-point est que l'application place un message dans une file d'attente locale lorsqu'elle envoie un message. Le système de messagerie et de mise en file d'attente sous-jacent détermine la file d'attente de destination à laquelle le message est envoyé. L'application de réception extrait le message de la file d'attente de destination.

une messagerie Publication / abonnement

Dans la messagerie Publier/S'abonner, il existe deux types d'application : le diffuseur et l'abonné.

Un *diffuseur* fournit des informations sous la forme de messages de publication. Lorsqu'un diffuseur de publications publie un message, il spécifie une rubrique, laquelle identifie l'objet des informations contenues dans le message.

Un *abonné* est un consommateur des informations publiées. Un abonné spécifie les rubriques qui l'intéressent en créant des abonnements.

Le système de publication/abonnement reçoit des publications des diffuseurs et des abonnements des abonnés. Il achemine les publications vers les abonnés. Un abonné reçoit uniquement les publications relatives aux rubriques auxquelles il s'est abonné.

Une caractéristique essentielle de la messagerie Publier/S'abonner est qu'un diffuseur identifie une rubrique lorsqu'il publie un message. Il n'identifie pas les abonnés. Si un message est publié sur une rubrique pour laquelle il n'y a aucun abonné, aucune application ne reçoit le message.

Une application peut être à la fois un diffuseur de publications et un abonné.

Modèle d'objet XMS

L'API XMS est une interface orientée objet. Le modèle d'objet XMS s'appuie sur le modèle d'objet JMS 1.1.

La liste suivante récapitule les principales classes XMS, ou types d'objet :

`ConnectionFactory`
Un objet `ConnectionFactory` encapsule un ensemble de paramètres pour une connexion. Une application utilise `ConnectionFactory` pour créer une connexion. Une application peut fournir les paramètres lors de l'exécution et créer un objet `ConnectionFactory`. Les paramètres de connexion peuvent également être stockés dans un référentiel d'objets gérés. Une application peut extraire un objet du référentiel et créer un objet `ConnectionFactory` à partir de celui-ci.

`Connection`
Un objet `Connection` encapsule une connexion active d'une application vers un serveur de messagerie. Une application utilise une connexion pour créer des sessions.

`Destination`
Une application envoie ou reçoit des messages via un objet `Destination`. Dans le domaine Publier/S'abonner, un objet `Destination` encapsule une rubrique et, dans le domaine point-à-point, un objet `Destination` encapsule une file d'attente. Une application peut fournir les paramètres pour créer un objet `Destination` lors de l'exécution. Elle peut également créer un objet `Destination` à partir d'une définition d'objet stockée dans un référentiel d'objets gérés.

`Session`
Un objet `Session` est un contexte à unité d'exécution unique pour l'envoi et la réception de messages. Une application utilise un objet `Session` pour créer des objets `Message`, `MessageProducer` et `MessageConsumer`.

`Message`
Un objet `Message` encapsule l'objet `Message` qu'une application envoie via un objet `MessageProducer` ou reçoit via un objet `MessageConsumer`.

`MessageProducer`
Un objet `MessageProducer` est utilisé par une application pour envoyer des messages vers une destination.

`MessageConsumer`
Un objet `MessageConsumer` est utilisé par une application pour recevoir des messages envoyés vers une destination.

La Figure 1, à la page 8 présente ces objets et leurs relations. Ce diagramme répertorie les principaux types d'objet XMS : ConnectionFactory, Connection, Session, MessageProducer, MessageConsumer, Message et Destination. Une application utilise une fabrique de connexions pour créer une connexion et utilise une connexion pour créer des sessions. L'application peut ensuite utiliser une session pour créer des messages, des expéditeurs de message et des consommateurs de message. L'application utilise un expéditeur de message pour envoyer des messages vers une destination et utilise un consommateur de message pour recevoir des messages envoyés vers une destination.

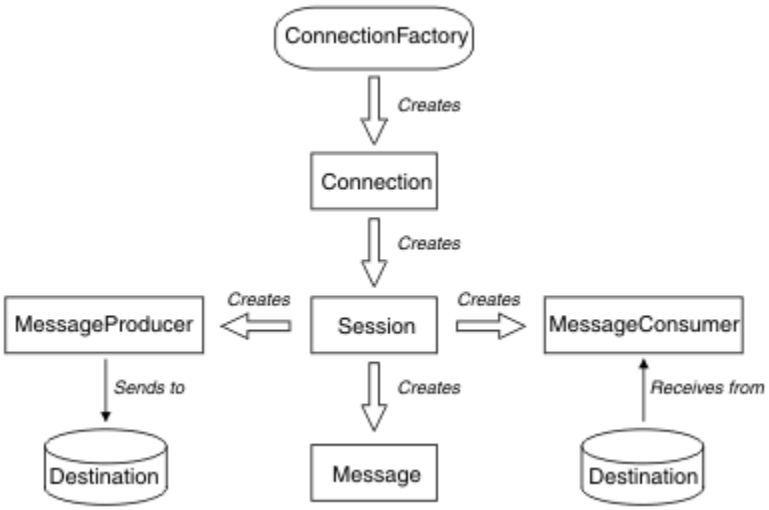

Figure 1. Objets XMS et leurs relations

Dans .NET, les classes XMS sont définies en tant qu'ensemble d'interfaces .NET. Lorsque vous codez des applications XMS .NET, vous devez seulement disposer des interfaces déclarées.

Le modèle d'objet XMS est basé sur les interfaces indépendantes du domaine décrites dans Java Message Service Specification, Version 1.1. Les classes spécifiques à un domaine, par exemple `Topic`, `TopicPublisher` et `TopicSubscriber`, ne sont pas fournies.

Attributs et propriétés des objets

Un objet XMS peut être caractérisé par des attributs et des propriétés, implémentés de différentes manières.

Attributs

Un attribut est une caractéristique d'objet, toujours présente et occupant de l'espace de stockage, même si aucune valeur ne lui est attribuée. Un attribut est donc similaire à une zone dans une structure de données de longueur fixe. Une des caractéristiques typiques des attributs est que chacun d'eux dispose de ses propres méthodes de définition et d'obtention de sa valeur.

Propriétés

Une propriété d'objet est présente et occupe de l'espace de stockage une fois que sa valeur a été définie. Une propriété ne peut pas être supprimée ou son espace de stockage récupéré une fois que sa valeur a été définie. Vous pouvez modifier cette valeur. XMS fournit un ensemble de méthodes génériques pour définir et obtenir des valeurs de propriété.

Concepts associés

Types primitifs XMS

XMS fournit des équivalents pour les huit types primitifs Java (byte, short, int, long, float, double, char et boolean). Ils permettent l'échange de messages entre XMS et JMS sans perte ni corruption de données.

Conversion implicite d'une valeur de propriété d'un type de données à un autre

Lorsqu'une application extrait la valeur d'une propriété, celle-ci peut être convertie par XMS dans un autre type de données. De nombreuses règles régissent les conversions prises en charge et la manière dont XMS les réalise.

Référence associée

Types de données d'éléments de données d'application

Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Objets gérés

Les objets gérés permettent d'administrer les paramètres de connexion utilisés à partir d'un référentiel central. Une application extrait des définitions d'objet depuis le référentiel central et les utilise pour créer des objets `ConnectionFactory` et `Destination`. Les objets gérés permettent de découpler les applications des ressources qu'elles utilisent lors de leur exécution.

Par exemple, des applications XMS peuvent être écrites et testées avec des objets gérés qui font référence à un ensemble de connexions et de destinations dans un environnement de test. Lorsque les applications sont déployées, les objets gérés peuvent être modifiés pour que les applications soient configurées de manière à faire référence aux connexions et aux destinations de l'environnement de production.

XMS prend en charge deux types d'objet géré :

- L'objet `ConnectionFactory`, utilisé par les applications pour établir la connexion initiale au serveur.
- L'objet `Destination`, utilisé par les applications pour spécifier la destination des messages envoyés et la source des messages reçus. Une destination est une rubrique ou une file d'attente sur le serveur auquel l'application se connecte.

L'outil d'administration **`JMSAdmin`** est intégré à IBM MQ. Il permet de créer et de gérer des objets gérés dans un référentiel central d'objets gérés.

Les objets gérés du référentiel peuvent être utilisés par les applications IBM MQ classes for JMS et XMS. Les applications XMS peuvent utiliser les objets `ConnectionFactory` et `Destination` pour se connecter à un IBM MQ gestionnaire de files d'attente. Un administrateur peut modifier les définitions d'objet conservées dans le référentiel sans affecter le code d'application.

Le diagramme suivant illustre la manière dont une application XMS utilise généralement les objets gérés. La partie gauche du diagramme présente un référentiel contenant des définitions d'objet ConnectionFactory et Destination gérés par l'intermédiaire d'une console d'administration. La partie droite du diagramme présente une application XMS qui recherche des définitions d'objet dans le référentiel, puis les utilise lors de la connexion à un serveur de messagerie.

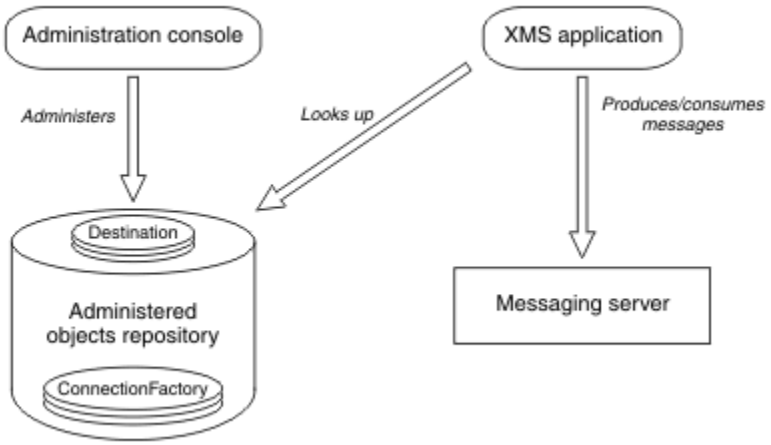

Figure 2. Utilisation standard d'objets gérés par une application XMS

Concepts associés

Utilisation des objets gérés
Les rubriques de cette section fournissent des informations sur les objets gérés. Les applications XMS peuvent extraire des définitions d'objet depuis un référentiel central d'objets gérés et les utiliser pour créer des fabriques de connexions et des destinations.

Types de référentiel d'objets gérés pris en charge

Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Tâches associées

Création d'objets gérés

La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Modèle de message XMS

Le modèle de message XMS est identique au modèle de message IBM MQ classes for JMS.

XMS implémente les mêmes zones d'en-tête et les mêmes propriétés de message que IBM MQ classes for JMS :

- Zones d'en-tête JMS. Ces zones portent un nom qui commence par le préfixe JMS.
- Propriétés définies par JMS. Ces zones contiennent des propriétés dont le nom commence par le préfixe JMSX.
- Propriétés définies par IBM. Ces zones contiennent des propriétés dont le nom commence par le préfixe JMS_IBM_.

En conséquence, les applications XMS peuvent échanger des messages avec des applications IBM MQ classes for JMS. Dans chaque message, certaines zones d'en-tête et certaines propriétés sont définies par l'application et d'autres, par XMS ou IBM MQ classes for JMS. Certaines zones définies par XMS ou IBM MQ classes for JMS le sont lors de l'envoi du message, d'autres lors de sa réception. Les zones d'en-tête et les propriétés sont propagées avec un message via un serveur de messagerie. Elles sont disponibles pour chaque application qui reçoit le message.

Installation de Message Service Client for .NET à l'aide de l'assistant d'installation

L'installation utilise le programme d'installation InstallShield X/Windows MSI. Vous avez le choix entre une installation complète ou personnalisée.

Pourquoi et quand exécuter cette tâche

Pour installer Message Service Client for .NET sur Windows, suivez cette procédure.

Procédure

1. Si vous procédez à l'installation à partir d'un SupportPac, suivez les étapes ci-après ; sinon, passez directement à l'étape «2», à la page 10.
   a) Sous Fenêtres, connectez-vous en tant qu'administrateur.
   b) Exécutez le programme d'installation dotNETClientsetup.exe.
2. Attendez que l'assistant d'installation s'ouvre et affiche le message suivant :

   ```
   Welcome to IBM Message Service Client for .NET installation wizard
   ```

   Cliquez sur **Suivant**.

   L'assistant peut vous demander de lire le contrat de licence.
3. Si vous êtes invité à lire le contrat de licence et que vous acceptez les dispositions du contrat de licence, cliquez sur **J'accepte les dispositions du contrat de licence** et cliquez ensuite sur le bouton **Suivant**.

L'assistant d'installation vous demande de choisir le type de configuration qui répond le mieux à vos besoins.

4. Sélectionnez le type de configuration dont vous avez besoin :
   - Pour installer toutes les fonctions du programme dans le répertoire d'installation par défaut, cliquez sur **Complète**.
   - Pour choisir les fonctions que vous voulez installer et spécifier l'endroit où vous voulez les installer, cliquez sur **Personnalisée**.
5. Cliquez sur **Suivant**.

   Si vous sélectionnez l'option d'installation complète, l'assistant affiche un message indiquant qu'il est prêt à commencer l'installation, comme décrit à l'étape «8», à la page 11. Si vous sélectionnez l'option d'installation personnalisée, l'assistant vous invite à sélectionner les fonctions que vous voulez installer ; vous devez procéder aux étapes «6», à la page 11 et «7», à la page 11 avant de passer à l'étape «8», à la page 11.
6. En cas d'installation personnalisée, cliquez sur une icône dans la liste des fonctions pour spécifier tout changement sur la manière dont vous voulez que les fonctions Message Service Client for .NET soient installées. Si vous ne voulez pas installer Message Service Client for .NET dans le répertoire proposé, choisissez-en un autre.

   Si vous choisissez d'installer Message Service Client for .NET dans un répertoire qui n'existe pas, l'assistant d'installation crée celui-ci pour vous.

   Si vous voulez développer des applications XMS, assurez-vous que la fonction **Outils de développement et exemples** est sélectionnée. Cette fonction fournit des modèles d'application, ainsi que les bibliothèques et d'autres fichiers requis pour compiler des applications .NET . Si vous ne sélectionnez pas cette fonction, seuls les fichiers requis pour exécuter des applications XMS sont installés.
7. Si vous utilisez l'option d'installation personnalisée, cliquez sur **Suivant** après avoir sélectionné les options de votre choix comme expliqué à l'étape «6», à la page 11.

   L'assistant d'installation affiche un message indiquant qu'il est prêt à commencer l'installation.
8. Cliquez sur **Installer** pour lancer l'installation.

   L'assistant d'installation affiche une barre indiquant la progression de l'installation. Attendez que la barre de progression indique 100 %. Lorsque l'installation est terminée, le message suivant s'affiche :

   ```
   The installation wizard has successfully installed IBM Message Service Client
   for .NET. Click Finish to exit the wizard.
   ```
9. Cliquez sur **Terminer** pour fermer l'assistant d'installation.

Résultats

Message Service Client for .NET a été installé. Il est prêt à être utilisé.

Que faire ensuite

Avant d'exécuter une application XMS, y compris les modèles d'application fournis avec XMS, vous devez configurer l'environnement du serveur de messagerie ; pour plus de détails, voir «Configuration de l'environnement d'un serveur de messagerie», à la page 13.

Concepts associés

service Web de recherche JNDI

Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Configuration de l'environnement d'un serveur de messagerie

Les rubriques de cette section expliquent comment configurer l'environnement d'un serveur de messagerie pour permettre aux applications XMS de se connecter à un serveur.

Utilisation des modèles d'application XMS
Utilisez les modèles d'application fournis avec XMS pour vérifier l'installation et la configuration du serveur de messagerie ; ces modèles vous aident également à générer vos propres applications. Les modèles fournissent une présentation des caractéristiques communes de chaque API.

Prérequis pour les applications XMS qui se connectent à WebSphere MQ

Certaines conditions s'appliquent si votre application XMS se connecte à WebSphere MQ.

Pour les applications qui se connectent à un gestionnaire de files d'attente WebSphere MQ, vous devez installer les bibliothèques client WebSphere MQ appropriées sur la machine que vous utilisez pour exécuter l'application XMS. Ces bibliothèques sont préinstallées sur des machines avec un gestionnaire de files d'attente local.

Pour XMS client for .NET, utilisez les bibliothèques client fournies avec IBM WebSphere MQ 7.0.1.0 ou version ultérieure. Il s'agit des classes *WebSphere MQ pour .NET* . Ils activent les connexions en mode client aux gestionnaires de files d'attente IBM WebSphere MQ 7.0, 6.0et 5.3 , ainsi que les connexions en mode liaison à un gestionnaire de files d'attente local, s'il s'agit également de IBM WebSphere MQ 7.0.1.0 ou d'une version ultérieure.

Microsoft .NET Framework 2.0 Redistributable Package doit être installé sur l'ordinateur sur lequel XMS doit être installé. Si ce module n'est pas disponible, l'installation de XMS échoue. Vous devez ensuite quitter la procédure d'installation, installer le package redistribuable Microsoft .NET Framework 2.0 sur votre ordinateur et réexécuter la procédure d'installation.

Sur le site de téléchargement Microsoft , vous devez rechercher dotnetfx.exe pour Microsoft .NET Framework 2.0 Redistributable Package (x86) et NetFx64.exe pour Microsoft .NET Framework 2.0 Redistributable Package (x64), selon le cas.

Concepts associés

Configuration de l'environnement d'un serveur de messagerie
Les rubriques de cette section expliquent comment configurer l'environnement d'un serveur de messagerie pour permettre aux applications XMS de se connecter à un serveur.

Désactivation de l'utilisation d'une nouvelle version de XMS pour les applications

Par défaut, lorsqu'une nouvelle version de XMS est installée, les applications qui utilisaient la version précédente basculent automatiquement vers la nouvelle version sans être recompilées.

Pourquoi et quand exécuter cette tâche

La fonction de coexistence de versions multiples permet de s'assurer que l'installation d'une version plus récente de XMS ne se substitue pas à la version précédente deXMS. En effet, plusieurs instances d'assemblagesXMS .NET similaires coexistent dans le GAC (Global Assembly Cache) avec des numéros de version différents. En interne, le GAC utilise un fichier de règles pour acheminer l'appel de l'application vers la version la plus récente de XMS. Les applications s'exécutent sans avoir besoin d'être recompilées et peuvent utiliser les nouvelles fonctionnalités de la version la plus récente de XMS .NET.

Toutefois, si une application doit utiliser une version plus ancienne deXMS, vous pouvez définir l'attribut `publisherpolicy` à no dans son fichier de configuration.

Remarque : Un fichier de configuration d'application est un fichier dont le nom est composé du nom du programme exécutable auquel il est associé, suivi du suffixe .config. Par exemple, le fichier de configuration d'application de text.exe se nommerait text.exe.config.

Toutefois, à tout moment, toutes les applications d'un système utilisent la même version de XMS .NET.

Configuration de l'environnement d'un serveur de messagerie

Les rubriques de cette section expliquent comment configurer l'environnement d'un serveur de messagerie pour permettre aux applications XMS de se connecter à un serveur.

Pour les applications qui se connectent à un gestionnaire de files d'attente IBM MQ, le client IBM MQ (ou le gestionnaire de files d'attente pour le mode liaisons) est requis.

Il n'y a aucun prérequis pour les applications qui utilisent une connexion en temps réel à un courtier.

Vous devez configurer l'environnement du serveur de messagerie avant d'exécuter des applications XMS, y compris les modèles d'application fournis avec XMS.

Cette section contient les rubriques suivantes :

Concepts associés

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Utilisation des modèles d'application XMS
Utilisez les modèles d'application fournis avec XMS pour vérifier l'installation et la configuration du serveur de messagerie ; ces modèles vous aident également à générer vos propres applications. Les modèles fournissent une présentation des caractéristiques communes de chaque API.

Tâches associées

Installation de Message Service Client for .NET à l'aide de l'assistant d'installation
L'installation utilise le programme d'installation InstallShield X/Windows MSI. Vous avez le choix entre une installation complète ou personnalisée.

Référence associée

Prérequis pour les applications XMS qui se connectent à WebSphere MQ
Certaines conditions s'appliquent si votre application XMS se connecte à WebSphere MQ.

Configuration du gestionnaire de files d'attente et du courtier pour une application se connectant à un gestionnaire de files d'attente IBM MQ

Cette section suppose que vous utilisez IBM WebSphere MQ 7.0.1 ou une version ultérieure. Pour pouvoir exécuter une application qui se connecte à un gestionnaire de files d'attente IBM MQ, vous devez configurer le gestionnaire de files d'attente. Pour une application de publication/abonnement, une configuration supplémentaire est requise si vous utilisez l'interface de publication/abonnement en file d'attente.

Avant de commencer

XMS fonctionne avec IBM Integration Bus ou WebSphere Message Broker 6.1 ou version ultérieure

Avant de démarrer cette tâche, vous devez effectuer les étapes suivantes :

- Assurez-vous que votre application a accès à un gestionnaire de files d'attente en cours d'exécution.

- Si l'application est de type publication/abonnement et utilise l'interface de publication/abonnement en file d'attente, assurez-vous que l'attribut "PSMODE" est défini sur "ENABLED" dans le gestionnaire de files d'attente.
- Assurez-vous que votre application utilise une fabrique de connexions dont les propriétés sont définies de manière appropriée pour la connexion au gestionnaire de files d'attente. Si votre application est de type publication/abonnement, assurez-vous que les propriétés de fabrique de connexions appropriées sont définies pour l'utilisation du courtier. Pour plus d'informations sur les propriétés d'une fabrique de connexions, voir «Propriétés de ConnectionFactory», à la page 182.

Pourquoi et quand exécuter cette tâche

Vous configurez le gestionnaire de files d'attente et le courtier pour exécuter des applications XMS de la même manière que vous configurez le gestionnaire de files d'attente et l'interface de publication / abonnement en file d'attente pour exécuter des applications IBM MQ JMS . Les étapes suivantes récapitulent la procédure à suivre.

Procédure

1. Dans le gestionnaire de files d'attente, créez les files d'attente dont votre application a besoin.

   Pour plus d'informations sur la création de files d'attente, voir Définition de files d'attente.

   Si votre application est une application de publication/abonnement et utilise une interface de publication/abonnement en file d'attente qui doit accéder aux files d'attente système IBM MQ classes for JMS, attendez jusqu'à l'étape 4a avant de créer les files d'attente.
2. Accordez à l'ID utilisateur associé à votre application le droit de connexion au gestionnaire de files d'attente, ainsi que les droits requis pour accéder aux files d'attente.

   Pour plus d'informations sur les droits, voir Sécurisation. Si votre application se connecte au gestionnaire de files d'attente en mode client, voir aussi Clients et serveurs.
3. Si votre application se connecte au gestionnaire de files d'attente en mode client, assurez-vous qu'un canal de connexion serveur est défini sur le gestionnaire de files d'attente et qu'un programme d'écoute est démarré.

   Il n'est pas nécessaire de répéter cette étape pour chaque application qui se connecte au gestionnaire de files d'attente. Une définition de canal de connexion serveur et un programme d'écoute peuvent prendre en charge toutes les applications qui se connectent en mode client.
4. Si votre application est une application de publication/abonnement et qu'elle utilise l'interface de publication/abonnement en file d'attente, effectuez les étapes ci-dessous.
   a) Dans le gestionnaire de files d'attente, créez les files d'attente système IBM MQ classes for JMS en exécutant le script des commandes MQSC fourni avec IBM MQ. Assurez-vous que l'ID utilisateur associé à IBM Integration Bus ou WebSphere Message Broker dispose des droits requis pour accéder aux files d'attente.

      Pour plus d'informations sur le script, voir Utilisation de IBM MQ classes for Java.

      Effectuez cette étape une seule fois pour le gestionnaire de files d'attente. Le même ensemble de files d'attente système IBM MQ classes for JMS peut prendre en charge toutes les applications XMS et IBM MQ classes for JMS se connectant au gestionnaire de files d'attente.

   b) Accordez à l'ID utilisateur associé à votre application le droits d'accès aux files d'attente système IBM MQ classes for JMS.

      Pour plus d'informations sur les droits dont l'ID utilisateur a besoin, voir Utilisation de IBM MQ classes for JMS.

   c) Pour un courtier IBM Integration Bus ou WebSphere Message Broker, créez et déployez un flux de messages pour servir la file d'attente sur laquelle les applications envoient les messages qu'elles publient.

      Le flux de messages de base comprend un noeud de traitement de messages MQInput pour lire les messages publiés et un noeud de traitement de messages Publication pour publier les messages.

Pour des informations sur la création et le déploiement d'un flux de messages, voir la documentation du produit IBM Integration Bus ou WebSphere Message Broker disponible depuis la Page Web de la bibliothèque de documentation du produit IBM Integration Bus.

Cette étape n'est pas utile si un flux de messages approprié est déjà déployé sur le courtier.

Résultats

Vous pouvez à présent démarrer votre application.

Tâches associées

Configuration d'un courtier pour une application utilisant une connexion en temps réel à un courtier
Avant de pouvoir exécuter une application utilisant une connexion en temps réel à un courtier, vous devez configurer ce dernier.

Configuration du bus d'intégration de services pour une application se connectant à WebSphere Application Server
Pour pouvoir exécuter une application qui se connecte à un bus d'intégration de services WebSphere Application Server service integration technologies, vous devez configurer l'intégration de services de la même façon que vous configurez le bus d'intégration de services pour exécuter des applications JMS utilisant le fournisseur de messagerie par défaut.

Configuration d'un courtier pour une application utilisant une connexion en temps réel à un courtier

Avant de pouvoir exécuter une application utilisant une connexion en temps réel à un courtier, vous devez configurer ce dernier.

Avant de commencer

Avant de démarrer cette tâche, vous devez effectuer les étapes suivantes :

- Assurez-vous que votre application a accès à un courtier en cours d'exécution.
- Assurez-vous que votre application utilise une fabrique de connexions dont les propriétés sont correctement définies pour une connexion en temps réel à un courtier. Pour plus d'informations sur les propriétés d'une fabrique de connexions, voir «Propriétés de ConnectionFactory», à la page 182.

Pourquoi et quand exécuter cette tâche

La configuration d'un courtier pour exécuter des applications XMS s'effectue de manière identique à la configuration d'un courtier pour exécuter des applications IBM MQ classes for JMS. Les étapes suivantes récapitulent la procédure à suivre :

Procédure

1. Créez et déployez un flux de messages pour lire des messages depuis le port TCP/IP sur lequel un courtier est à l'écoute et publiez les messages.

   Pour ce faire, vous pouvez procéder selon l'une des manières suivantes :

   - Créer un flux de messages qui contient un noeud de traitement de message **Real-timeOptimizedFlow**.
   - Créer un flux de messages qui contient un noeud de traitement de message **Real-timeInput** et un noeud de traitement de message Publication.

   Vous devez configurer le noeud **Real-timeOptimizedFlow** ou **Real-timeInput** pour écouter sur le port utilisé pour les connexions en temps réel. Dans XMS, le numéro de port par défaut pour les connexions en temps réel est 1506.

   Cette étape n'est pas utile si un flux de messages approprié est déjà déployé sur le courtier.
2. Si vous avez besoin que des messages soient livrés à votre application à l'aide de IBM MQ classes for JMS, configurez le courtier de manière à activer la multidiffusion. Configurez les rubriques pour

lesquelles la multidiffusion doit être activée, en spécifiant une qualité de service fiable pour ces rubriques nécessitant un mode multidiffusion fiable.

3. Si votre application fournit un ID utilisateur et un mot de passe lorsqu'elle se connecte au courtier, et si vous souhaitez que le courtier authentifie votre application par l'intermédiaire de ces informations, configurez le serveur et le courtier de manière à activer une authentification simple par mot de passe de type telnet.

Résultats

Vous pouvez à présent démarrer votre application.

Tâches associées

Configuration du gestionnaire de files d'attente et du courtier pour une application se connectant à un gestionnaire de files d'attente IBM MQ

Cette section suppose que vous utilisez IBM WebSphere MQ 7.0.1 ou une version ultérieure. Pour pouvoir exécuter une application qui se connecte à un gestionnaire de files d'attente IBM MQ, vous devez configurer le gestionnaire de files d'attente. Pour une application de publication/abonnement, une configuration supplémentaire est requise si vous utilisez l'interface de publication/abonnement en file d'attente.

Configuration du bus d'intégration de services pour une application se connectant à WebSphere Application Server

Pour pouvoir exécuter une application qui se connecte à un bus d'intégration de services WebSphere Application Server service integration technologies, vous devez configurer l'intégration de services de la même façon que vous configurez le bus d'intégration de services pour exécuter des applications JMS utilisant le fournisseur de messagerie par défaut.

Configuration du bus d'intégration de services pour une application se connectant à WebSphere Application Server

Pour pouvoir exécuter une application qui se connecte à un bus d'intégration de services WebSphere Application Server service integration technologies, vous devez configurer l'intégration de services de la même façon que vous configurez le bus d'intégration de services pour exécuter des applications JMS utilisant le fournisseur de messagerie par défaut.

Avant de commencer

Avant de démarrer cette tâche, vous devez effectuer les étapes suivantes :

- Assurez-vous qu'un bus de messagerie est créé et que votre serveur est ajouté au bus en tant que membre de bus.
- Assurez-vous que votre application a accès à un bus d'intégration de services qui contient au moins un moteur de messagerie en cours d'exécution.
- Si le mode HTTP est requis, un canal de transport entrant de moteur de messagerie HTTP doit être défini. Par défaut, les canaux pour SSL et TCP sont définis au cours de l'installation du serveur.
- Assurez-vous que votre application utilise une fabrique de connexions dont les propriétés sont définies de manière appropriée pour la connexion au bus d'intégration de services à l'aide d'un serveur d'amorçage. Les informations minimales requises sont :
  - Le noeud final du fournisseur, qui décrit l'emplacement et le protocole à utiliser lors de la négociation d'une connexion au serveur de messagerie (c'est-à-dire, le serveur d'amorçage). Dans son format le plus simple, pour un serveur installé avec des paramètres par défaut, le noeud final peut être défini par le nom d'hôte du serveur.
  - Le nom du bus par lequel les messages sont envoyés.

  Pour plus d'informations sur les propriétés d'une fabrique de connexions, voir «Propriétés de ConnectionFactory», à la page 182.

Pourquoi et quand exécuter cette tâche

Les espaces de rubrique ou de file d'attente dont vous avez besoin doivent être définis. Par défaut, un espace de rubrique appelé Default.Topic.Space est défini au cours de l'installation du serveur mais, si vous avez besoin d'espaces de rubrique supplémentaires, vous devez les créer vous-même. Vous n'avez pas besoin de prédéfinir des rubriques individuelles dans un espace de rubriques, car le serveur instancie de manière dynamique les rubriques individuelles si nécessaire.

Les étapes suivantes récapitulent la procédure à suivre.

Procédure

1. Créez les files d'attente dont votre application a besoin pour la messagerie point-à-point.
2. Créez les espaces de sujet supplémentaires dont votre application a besoin pour la messagerie de publication/abonnement.

Résultats

Vous pouvez à présent démarrer votre application.

Tâches associées

Configuration du gestionnaire de files d'attente et du courtier pour une application se connectant à un gestionnaire de files d'attente IBM MQ

Cette section suppose que vous utilisez IBM WebSphere MQ 7.0.1 ou une version ultérieure. Pour pouvoir exécuter une application qui se connecte à un gestionnaire de files d'attente IBM MQ, vous devez configurer le gestionnaire de files d'attente. Pour une application de publication/abonnement, une configuration supplémentaire est requise si vous utilisez l'interface de publication/abonnement en file d'attente.

Configuration d'un courtier pour une application utilisant une connexion en temps réel à un courtier

Avant de pouvoir exécuter une application utilisant une connexion en temps réel à un courtier, vous devez configurer ce dernier.

Utilisation des modèles d'application XMS

Utilisez les modèles d'application fournis avec XMS pour vérifier l'installation et la configuration du serveur de messagerie ; ces modèles vous aident également à générer vos propres applications. Les modèles fournissent une présentation des caractéristiques communes de chaque API.

Concepts associés

service Web de recherche JNDI

Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Configuration de l'environnement d'un serveur de messagerie

Les rubriques de cette section expliquent comment configurer l'environnement d'un serveur de messagerie pour permettre aux applications XMS de se connecter à un serveur.

Tâches associées

Installation de Message Service Client for .NET à l'aide de l'assistant d'installation

L'installation utilise le programme d'installation InstallShield X/Windows MSI. Vous avez le choix entre une installation complète ou personnalisée.

Modèles d'application

Les modèles d'application fournissent une vue d'ensemble des caractéristiques communes de chaque API. Vous pouvez les utiliser pour vérifier votre installation et votre configuration de serveur de messagerie, ou pour vous aider à construire vos propres applications.

Si vous avez besoin d'aide pour créer vos propres applications, vous pouvez utiliser les modèles d'application comme base de départ. La version source et la version compilée sont fournies pour chaque application. Utilisez le code source exemple pour identifier les étapes clé permettant de créer chaque objet requis pour votre application (ConnectionFactory, Connection, Session, Destination, mais aussi un expéditeur et/ou un consommateur) et de définir les propriétés spécifiques dont vous avez besoin pour spécifier la manière dont l'application doit fonctionner. Pour plus d'informations, voir la section «Ecriture d'applications XMS», à la page 20. Les exemples sont susceptibles de changer dans les éditions ultérieures de XMS.

Le tableau suivant présente les trois ensembles de modèles d'application (un par API) fournis avec XMS.

Nom du modèle	Description
SampleConsumerCS	Application consommateur de message qui prend les messages d'une file d'attente ou s'abonne à une rubrique.
SampleProducerCS	Application expéditeur de message qui fournit des messages à une file d'attente ou à une rubrique.
SampleConfigCS	Application de configuration que vous pouvez utiliser pour créer un référentiel d'objets gérés à partir de fichiers. L'application contient une fabrique de connexions et une destination pour vos paramètres de connexion spécifiques. Ce référentiel d'objets gérés peut ensuite être utilisé avec chaque modèle d'application consommateur et expéditeur.

Les modèles qui prennent en charge les mêmes fonctions dans les diverses API présentent des différences au niveau de la syntaxe.

- Les modèles d'application consommateur et expéditeur prennent en charge les fonctions suivantes :
  - Connexion à IBM MQ, IBM Integration Bus (via une connexion en temps réel à un courtier) et à un WebSphere Application Server service integration bus
  - Recherche dans le référentiel d'objets gérés via l'interface de contexte initial
  - Connexion aux files d'attente (IBM MQ et WebSphere Application Server service integration bus) et aux rubriques (IBM MQ, connexion en temps réel à un courtier, et à WebSphere Application Server service integration bus)
  - Messages de base, d'octets, de mappe, d'objet, de flux et texte
- Le modèle d'application consommateur de message prend en charge les modes de réception synchrone et asynchrone, et les instructions SQL Selector.
- Le modèle d'application expéditeur de message prend en charge les modes de livraison permanent et non permanent.

Modes d'exploitation

Les modèles peuvent fonctionner dans deux modes :

Mode simple
Vous pouvez exécuter le modèle avec un minimum d'entrée utilisateur.

Mode avancé
Vous pouvez personnaliser plus en profondeur la manière dont le modèle fonctionne.

Tous les modèles sont compatibles et peuvent fonctionner quelle que soit la langue.

Concepts associés

Génération de vos propres applications
Vous pouvez générer vos propres applications comme vous générez les modèles d'application.

Tâches associées

Exécution des modèles d'application

Vous pouvez exécuter les modèles d'application .NET de manière interactive en mode simple ou avancé, ou de manière non interactive à l'aide des fichiers de réponse générés automatiquement ou personnalisés.

Génération des modèles d'application .NET
Lorsque vous générez un modèle d'application .NET, une version exécutable du modèle choisi est créée.

Exécution des modèles d'application

Vous pouvez exécuter les modèles d'application .NET de manière interactive en mode simple ou avancé, ou de manière non interactive à l'aide des fichiers de réponse générés automatiquement ou personnalisés.

Avant de commencer

Avant d'exécuter un des modèles d'application fournis, vous devez commencer par configurer l'environnement du serveur de messagerie, de sorte que les applications puissent se connecter à un serveur. Voir «Configuration de l'environnement d'un serveur de messagerie», à la page 13.

Procédure

Pour exécuter un modèle d'application .NET, effectuez les étapes suivantes :

Conseil : Lorsque vous exécutez un exemple d'application, entrez? à tout moment pour obtenir de l'aide sur ce qu'il faut faire ensuite.

1. Sélectionnez le mode dans lequel vous voulez exécuter le modèle d'application.

   Entrez `Advanced` ou `Simple`.
2. Répondez aux questions.

   Pour sélectionner la valeur par défaut, qui apparaît entre crochets à la fin de la question, appuyez sur Entrée. Pour sélectionner une autre valeur, tapez celle-ci et appuyez sur Entrée.

   Voici un exemple de question :

   ```
   Enter connection type [wpm]:
   ```

   Dans ce cas, la valeur par défaut est `wpm` (connexion à un WebSphere Application Server service integration bus).

Résultats

Lorsque vous exécutez les modèles d'application, des fichiers de réponses sont générés automatiquement dans le répertoire de travail en cours. Les noms de fichier de réponses sont au format *`connection_type-sample_type`*`.rsp`; par exemple, `wpm-producer.rsp`. Si besoin, vous pouvez utiliser les fichiers de réponses générés pour exécuter à nouveau les modèles d'application avec les mêmes options, de sorte que vous n'avez pas besoin de les entrer une nouvelle fois.

Concepts associés

Modèles d'application
Les modèles d'application fournissent une vue d'ensemble des caractéristiques communes de chaque API. Vous pouvez les utiliser pour vérifier votre installation et votre configuration de serveur de messagerie, ou pour vous aider à construire vos propres applications.

Tâches associées

Génération des modèles d'application .NET
Lorsque vous générez un modèle d'application .NET, une version exécutable du modèle choisi est créée.

Génération des modèles d'application .NET

Lorsque vous générez un modèle d'application .NET, une version exécutable du modèle choisi est créée.

Avant de commencer

Installez le compilateur approprié. Cette tâche suppose que vous avez installé Microsoft Visual Studio 2012 et que vous savez l'utiliser.

Procédure

Pour générer un modèle d'application .NET, effectuez les étapes suivantes :

1. Cliquez sur le fichier de solution `Samples.sln` fourni avec les modèles .NET.
2. Cliquez avec le bouton droit de la souris sur la solution `Exemples` dans la fenêtre Explorateur de solutions et sélectionnez **Générer**.

Résultats

Un programme exécutable est créé dans le sous-dossier approprié de l'exemple, `bin/Debug` ou `bin/Release`, en fonction de la configuration que vous avez choisie. Ce programme a le même nom que le dossier, avec le suffixe CS. Par exemple, si vous générez la version C# de l'exemple d'application de l'expéditeur de messages, `SampleProducerCS.exe` est créé dans le dossier `SampleProducer`.

Concepts associés

Modèles d'application

Les modèles d'application fournissent une vue d'ensemble des caractéristiques communes de chaque API. Vous pouvez les utiliser pour vérifier votre installation et votre configuration de serveur de messagerie, ou pour vous aider à construire vos propres applications.

«Génération de vos propres applications», à la page 44

Vous pouvez générer vos propres applications comme vous générez les modèles d'application.

Tâches associées

Exécution des modèles d'application

Vous pouvez exécuter les modèles d'application .NET de manière interactive en mode simple ou avancé, ou de manière non interactive à l'aide des fichiers de réponse générés automatiquement ou personnalisés.

Développement d'applications XMS

Les rubriques de cette section fournissent des informations qui peuvent vous être utiles si vous écrivez des applications XMS.

Pour plus d'informations sur l'écriture d'applications XMS, reportez-vous aux rubriques suivantes :

Ecriture d'applications XMS

Les rubriques de cette section contiennent des informations d'aide à l'écriture d'applications XMS.

Cette section présente les concepts généraux d'écriture d'applications XMS. Voir aussi «Ecriture d'applications XMS .NET», à la page 45 pour des informations spécifiques sur la création d'applications .NET.

Cette section contient les rubriques suivantes :

- «Types primitifs XMS», à la page 40
- «Conversion implicite d'une valeur de propriété d'un type de données à un autre», à la page 41
- «Éléments itératifs», à la page 43
- «Identificateurs de jeu de caractères codés», à la page 44
- «Codes d'erreur et d'exception XMS», à la page 44
- «Génération de vos propres applications», à la page 44

Concepts associés

Ecriture d'applications XMS .NET
Les rubriques de cette section fournissent des informations qui peuvent vous être utiles si vous écrivez des applications XMS .NET.

Référence associée

.NET interfaces
Cette section documente les interfaces de classe .NET , ainsi que leurs propriétés et leurs méthodes.

Le modèle d'unités d'exécution

Des règles générales déterminent comment une application à unités d'exécution multiples peut utiliser des objets XMS.

- Seuls les objets des types suivants peuvent être utilisés simultanément sur des unités d'exécution différentes :
  - ConnectionFactory
  - Connexion
  - ConnectionMetaData
  - Destination
- Un objet Session peut être utilisé sur une seule unité d'exécution à un moment donné.

Les exceptions à ces règles sont indiquées par des entrées portant le libellé "Contexte d'unité d'exécution" dans les définitions d'interface des méthodes dans «Message Service Clients for .NET», à la page 92.

Concepts associés

Cas d'erreur pouvant être gérés lors de l'exécution
Les codes retour des appels API sont des cas d'erreur qui peuvent être gérés lors de l'exécution. La manière dont vous gérez ce type d'erreurs varie selon que vous utilisez l'API C ou C++.

Objets ConnectionFactories et Connection

Un objet ConnectionFactory fournit un modèle utilisé par une application pour créer un objet Connection. L'application utilise l'objet Connection pour créer un objet Session.

Pour .NET, l'application XMS utilise tout d'abord un objet XMSFactoryFactory pour extraire une référence à un objet ConnectionFactory approprié pour le type de protocole requis. Cet objet ConnectionFactory peut ensuite produire des connexions uniquement pour ce type de protocole.

L'application XMS peut créer plusieurs connexions, et une application à plusieurs unités d'exécution peut utiliser un objet Connection unique simultanément sur plusieurs unités d'exécution. Un objet Connection encapsule une connexion de communications entre une application et un serveur de messagerie.

Une connexion a plusieurs objectifs :

- Lorsqu'une application crée une connexion, elle peut être authentifiée.
- Une application peut associer un identificateur de client unique à une connexion. L'identificateur de client est utilisé pour la prise en charge d'abonnements durables dans le domaine de publication/abonnement. L'identificateur de client peut être défini de deux manières :

La meilleure manière d'affecter un identificateur de client de connexions consiste à configurer un objet ConnectionFactory propre au client et à l'affecter de manière transparente à la connexion qu'il crée.

Une autre manière d'affecter un identificateur de client est d'utiliser une valeur propre au client définie sur l'objet Connection. Cette valeur ne se substitue pas à l'identificateur qui a été configuré de manière administrative. Elle est fournie au cas où il n'existe aucun identificateur spécifié de manière administrative. Si un identificateur spécifié de manière administrative existe, une tentative de substitution par une valeur propre au fournisseur génère l'émission d'une exception. Si une application définit un identificateur de manière explicite, elle doit le faire immédiatement après avoir créé la connexion ou avant toute autre action sur la connexion ; sinon, une exception est émise.

Généralement, l'application XMS crée une connexion, une ou plusieurs sessions, des expéditeurs de message et des consommateurs de message.

La création d'une connexion est relativement coûteuse en termes de ressources système, car cela implique d'établir une connexion de communications, mais aussi éventuellement d'authentifier l'application.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Mode démarré et arrêt d'une connexion

Une connexion peut fonctionner en mode démarré ou arrêté.

Lorsqu'une application crée une connexion, celle-ci est en mode arrêté. Dans ce cas, l'application peut initialiser des sessions et envoyer des messages, mais ne peut pas en recevoir, que ce soit en mode synchrone ou asynchrone.

Une application peut démarrer une connexion en appelant la méthode `Start Connection` . Lorsque la connexion est en mode démarré, l'application peut envoyer et recevoir des messages. L'application peut ensuite arrêter et redémarrer la connexion en appelant les méthodes Stop Connection et `Start Connection`.

Concepts associés

Fermeture de connexion
Une application ferme une connexion en appelant la méthode Close Connection.

Traitement des exceptions
Si une application utilise une connexion uniquement pour consommer des messages en mode asynchrone, elle est avertie des problèmes liés à la connexion via un programme d'écoute des exceptions.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Fermeture de connexion

Une application ferme une connexion en appelant la méthode Close Connection.

Lorsqu'une application ferme une connexion, XMS effectue les actions suivantes :

- Il ferme toutes les sessions associées à la connexion et supprime certains objets associés à ces sessions. Pour plus d'informations sur les objets supprimés, voir «Suppression d'objet», à la page 39. Dans le même temps, XMS annule toutes les transactions en cours dans les sessions.
- Il met fin à la connexion de communications avec le serveur de messagerie.
- Il libère la mémoire et les autres ressources internes utilisées par la connexion.

XMS n'accuse pas réception des messages pour lesquels il n'est pas parvenu à effectuer cette opération avant la fermeture de la connexion. Pour plus d'informations sur l'accusé de réception des messages, voir «Accusé de réception de message», à la page 26.

Concepts associés

Mode démarré et arrêt d'une connexion
Une connexion peut fonctionner en mode démarré ou arrêté.

Traitement des exceptions
Si une application utilise une connexion uniquement pour consommer des messages en mode asynchrone, elle est avertie des problèmes liés à la connexion via un programme d'écoute des exceptions.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Traitement des exceptions

Si une application utilise une connexion uniquement pour consommer des messages en mode asynchrone, elle est avertie des problèmes liés à la connexion via un programme d'écoute des exceptions.

Les exceptions XMS .NET sont toutes dérivées de System.Exception. Pour plus d'informations, voir la section «Traitement des erreurs dans .NET», à la page 49.

Concepts associés

Mode démarré et arrêt d'une connexion
Une connexion peut fonctionner en mode démarré ou arrêté.

Fermeture de connexion
Une application ferme une connexion en appelant la méthode Close Connection.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Connexion à un bus d'intégration de services

Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Le protocole HTTP peut être utilisé dans les situations où une connexion TCP/IP directe n'est pas possible. Une situation courante est la communication via un pare-feu, par exemple lorsque deux entreprises échangent des messages. L'utilisation du protocole HTTP pour communiquer via un pare-feu est souvent appelée *tunnellisation HTTP.* Toutefois, cette méthode est par nature plus lente que l'utilisation d'une connexion TCP/IP directe, car les en-têtes HTTP s'ajoutent de manière significative à la quantité de données transférées ; de plus, le protocole HTTP nécessite plus de flux de communication que TCP/IP.

Pour créer une connexion TCP/IP, une application peut utiliser une fabrique de connexions dont la propriétéXMSC_WPM_TARGET_TRANSPORT_CHAIN est définie sur XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC. Il s'agit de la valeur par défaut de la propriété. Si la création de la connexion abouti, la propriétéXMSC_WPM_CONNECTION_PROTOCOL de la connexion est définie sur XMSC_WPM_CP_TCP.

Pour créer une connexion qui utilise le protocole HTTP, une application doit utiliser une fabrique de connexions dont la propriété XMSC_WPM_TARGET_TRANSPORT_CHAIN est définie sur le nom d'une chaîne de transport de communications entrante, c'est-à-dire configurée pour utiliser un canal de transport HTTP. Si la création de la connexion abouti, la propriété XMSC_WPM_CONNECTION_PROTOCOL de la connexion est définie sur XMSC_WPM_CP_HTTP. Pour des informations sur la configuration des chaînes de transport, voir Configuration des chaînes de transport dans la documentation du produit WebSphere Application Server.

Une application a un choix similaire de protocoles de communication pour une connexion à un serveur d'amorçage. La propriété XMSC_WPM_PROVIDER_ENDPOINTS d'une fabrique de connexions est une séquence de une ou plusieurs adresses de noeud final de serveurs d'amorçage. Le composant de chaîne de transport d'amorçage de chaque adresse de noeud final peut être XMSC_WPM_BOOTSTRAP_TCP, pour une connexion TCP/IP à un serveur d'amorçage, ou XMSC_WPM_BOOTSTRAP_HTTP pour une connexion via HTTP.

Concepts associés

Mode démarré et arrêt d'une connexion
Une connexion peut fonctionner en mode démarré ou arrêté.

Fermeture de connexion
Une application ferme une connexion en appelant la méthode Close Connection.

Traitement des exceptions
Si une application utilise une connexion uniquement pour consommer des messages en mode asynchrone, elle est avertie des problèmes liés à la connexion via un programme d'écoute des exceptions.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Sessions

Une session est un contexte à unité d'exécution unique pour l'envoi et la réception de messages.

Une application peut utiliser une session pour créer des messages, des expéditeurs de messages, des consommateurs de messages, des navigateurs de files d'attente et des destinations temporaires. Une application peut également utiliser une session pour exécuter des transactions locales.

Une application peut créer des sessions multiples, où chaque session produit et consomme des messages indépendamment des autres sessions. Si deux consommateurs de message dans des sessions distinctes (ou même dans la même session) s'abonnent à la même rubrique, chacun reçoit une copie de tout message publié sur ce sujet.

Contrairement à un objet Connexion, un objet Session ne peut pas être utilisé simultanément sur plusieurs unités d'exécution. Seule la méthode Close Session d'un objet Session peut être appelée à

partir d'une unité d'exécution autre que celle utilisée au même moment par l'objet Session. La méthode Close Session met fin à une session et libère les ressources système qui y étaient allouées.

Si une application doit traiter des messages simultanément sur plusieurs unités d'exécution, l'application doit créer une session sur chaque unité d'exécution, puis utiliser cette session pour toute opération d'envoi ou de réception au sein de cette unité d'exécution.

Sessions transactionnelles

Les applications XMS peuvent exécuter des transactions locales. Une *transaction locale* est une transaction qui implique uniquement les modifications apportées aux ressources du gestionnaire de files d'attente ou du bus d'intégration de services auquel l'application est connectée.

Les informations de cette rubrique sont pertinentes uniquement si une application se connecte à un gestionnaire de files d'attente IBM MQ ou à un bus d'intégration de services WebSphere Application Server. Elles ne le sont pas pour une connexion en temps réel à un courtier.

Pour exécuter des transactions locales, une application doit tout d'abord créer une session transactionnelle en appelant la méthode Create Session d'un objet Connection, et en spécifiant sous forme de paramètre que la session est transactionnelle. Par la suite, tous les messages envoyés et reçus au cours de la session sont groupés dans une séquence de transactions. Une transaction se termine lorsque l'application valide ou annule les messages qu'il a envoyés et reçus depuis le début de la transaction.

Pour valider une transaction, une application appelle la méthode Commit de l'objet Session. Lorsqu'une transaction est validée, tous les messages envoyés au cours de cette transaction deviennent disponibles pour la distribution à d'autres applications ; de même, un accusé de réception est envoyé pour tous les messages reçus au cours de cette transaction, de sorte que le serveur de messagerie n'essaie pas de les distribuer à nouveau à l'application. Dans le domaine point-à-point, le serveur de messagerie retire également les messages reçus de leurs files d'attente.

Pour annuler une transaction, une application appelle la méthode Rollback de l'objet Session. Lorsqu'une transaction est annulée, tous les messages envoyés au cours de cette transaction sont supprimés du serveur de messagerie, et tous les messages reçus au cours de cette transaction sont à nouveau disponible pour la distribution. Dans le domaine point-à-point, les messages reçus sont replacés dans leurs filesd'attente et les autres applications peuvent à nouveau les voir.

Une nouvelle transaction commence automatiquement lorsqu'une application crée une session transactionnelle ou appelle la méthode Commit ou Rollback. Une session transactionnelle a donc toujours une transaction active.

Lorsqu'une application ferme une session transactionnelle, une annulation implicite se produit. Lorsqu'une application ferme une connexion, une annulation implicite de toutes les sessions transactionnelles de la connexion se produit.

Une transaction est entièrement contenue dans une session transactionnelle. Une transaction ne peut pas s'étendre à d'autres sessions. Cela signifie qu'une application ne peut pas envoyer ou recevoir des messages dans plusieurs sessions transactionnelles, puis valider ou annuler toutes ces actions comme une transaction unique.

Concepts associés

Accusé de réception de message

Chaque session non transactionnelle possède un mode d'accusé de réception qui détermine la façon dont l'application accuse réception des messages. Trois modes d'accusé de réception sont disponibles, et le choix du mode d'accusé de réception a un impact sur la conception de l'application.

une distribution asynchrone de messages

XMS utilise une unité d'exécution pour gérer les distributions asynchrones de messages pour une session. Cela signifie qu'une seule fonction de programme d'écoute de message ou une méthode `onMessage()` peut être exécutée à la fois.

Distribution synchrone des messages

Les messages sont distribués de manière synchrone à une application lorsque celle-ci utilise les méthodes Receive des objets MessageConsumer.

Mode de livraison des messages
XMS prend en charge deux modes de livraison de messages.

Accusé de réception de message

Chaque session non transactionnelle possède un mode d'accusé de réception qui détermine la façon dont l'application accuse réception des messages. Trois modes d'accusé de réception sont disponibles, et le choix du mode d'accusé de réception a un impact sur la conception de l'application.

Les informations de cette rubrique sont pertinentes uniquement si une application se connecte à un gestionnaire de files d'attente IBM MQ ou à un bus d'intégration de services WebSphere Application Server. Elles ne le sont pas pour une connexion en temps réel à un courtier.

XMS utilise le même mécanisme d'accusé de réception de messages que JMS.

Si une session n'est pas transactionnelle, la façon dont l'application accuse réception des messages dépend du mode d'accusé de réception de la session. Les trois modes d'accusé de réception sont décrits dans les paragraphes suivants :

XMSC_AUTO_ACKNOWLEDGE
La session accuse automatiquement réception de chaque message reçu par l'application.

Si des messages sont distribués de manière synchrone à l'application, la session accuse réception chaque fois qu'un appel Receive aboutit.

Si l'application reçoit un message, mais qu'un incident empêche l'émission de l'accusé de réception, le message est à nouveau disponible pour la distribution. L'application doit donc être capable de gérer un message redistribué.

XMSC_DUPS_OK_ACKNOWLEDGE
La session accuse réception des messages reçus par l'application au moment de leur sélection.

Ce mode d'accusé de réception réduit le volume de travail que la session doit accomplir, mais si un incident empêche d'accuser réception du message, il se peut que plusieurs messages deviennent disponibles pour une nouvelle distribution. L'application doit donc être capable de gérer les messages redistribués.

XMSC_CLIENT_ACKNOWLEDGE
L'application accuse réception des messages qu'elle reçoit en appelant la méthode Acknowledge de la classe Message.

L'application peut accuser réception de chaque message individuellement ou recevoir un lot de messages et appeler la méthode Acknowledge seulement pour le dernier message reçu. Dans ce cas, l'accusé de réception est émis pour tous les messages reçus depuis l'appel précédent à cette méthode.

Conjointement avec ces modes d'accusé de réception, une application peut arrêter et redémarrer la distribution des messages dans une session en appelant la méthode Recover de la classe Session. Les messages dont l'accusé de réception n'a pas été envoyé sont redistribués. Toutefois, il est possible qu'ils ne soient pas redistribués dans le même ordre que la fois précédente. Entre temps, des messages de priorité plus élevée peuvent être arrivés, et certains messages originaux peuvent avoir expiré. Dans le domaine point-à-point, certains messages originaux peuvent avoir été consommés par une autre application.

Une application peut déterminer si un message est en cours de redistribution en examinant le contenu de la zone d'en-tête JMSRedelivered de celui-ci. L'application effectue cette opération en appelant la méthode Get JMSRedelivered de la classe Message.

Concepts associés
Sessions transactionnelles
Les applications XMS peuvent exécuter des transactions locales. Une *transaction locale* est une transaction qui implique uniquement les modifications apportées aux ressources du gestionnaire de files d'attente ou du bus d'intégration de services auquel l'application est connectée.

une distribution asynchrone de messages

XMS utilise une unité d'exécution pour gérer les distributions asynchrones de messages pour une session. Cela signifie qu'une seule fonction de programme d'écoute de message ou une méthode onMessage() peut être exécutée à la fois.

Distribution synchrone des messages
Les messages sont distribués de manière synchrone à une application lorsque celle-ci utilise les méthodes Receive des objets MessageConsumer.

Mode de livraison des messages
XMS prend en charge deux modes de livraison de messages.

une distribution asynchrone de messages

XMS utilise une unité d'exécution pour gérer les distributions asynchrones de messages pour une session. Cela signifie qu'une seule fonction de programme d'écoute de message ou une méthode onMessage() peut être exécutée à la fois.

Si, dans une session, plusieurs consommateurs de message reçoivent des messages de façon asynchrone et qu'une fonction de programme d'écoute de message ou une méthode onMessage() distribue un message à un consommateur, les consommateurs qui attendent le même message doivent patienter. Les autres messages qui attendent d'être distribués dans la session doivent également patienter.

Si une application requiert une distribution simultanée de messages, vous devez créer plusieurs sessions de sorte queXMS utilise plusieurs unités d'exécution pour gérer la distribution asynchrone. De cette manière, plusieurs fonctions de programme d'écoute de message ou plusieurs méthodes onMessage() peuvent être exécutées simultanément.

Une session n'est pas rendue asynchrone par l'affectation d'un programme d'écoute de message à un consommateur. Une session devient asynchrone uniquement lorsque la méthode Connection.Start est appelée. Tous les appels synchrones sont autorisés jusqu'à ce que la méthode Connection.Start soit appelée. La distribution de messages aux consommateurs commence lorsque Connection.Start est appelée.

Les appels synchrones, par exemple une création de consommateur ou de fournisseur, doivent être faits sur une session asynchrone et Connection.Stop doit être appelée. Une session peut être reprise par appel de la méthode Connection.Start pour lancer la distribution des messages. La seule exception est l'unité d'exécution de distribution de messages Session, qui est l'unité d'exécution en charge de la distribution des messages à la fonction de rappel. Cette unité d'exécution peut lancer n'importe quel appel sur une session (à l'exception de l'appel Close) dans la fonction de rappel de message.

Remarque : En mode non géré, l'appel MQDISC dans une fonction de rappel n'est pas pris en charge par le client IBM MQ .NET. Ainsi, l'application client ne peut pas créer ou fermer des sessions dans le rappel MessageListener en mode de réception asynchrone. Créez et supprimez la session en dehors de la méthode MessageListener.

Concepts associés

Sessions transactionnelles
Les applications XMS peuvent exécuter des transactions locales. Une *transaction locale* est une transaction qui implique uniquement les modifications apportées aux ressources du gestionnaire de files d'attente ou du bus d'intégration de services auquel l'application est connectée.

Accusé de réception de message
Chaque session non transactionnelle possède un mode d'accusé de réception qui détermine la façon dont l'application accuse réception des messages. Trois modes d'accusé de réception sont disponibles, et le choix du mode d'accusé de réception a un impact sur la conception de l'application.

Distribution synchrone des messages
Les messages sont distribués de manière synchrone à une application lorsque celle-ci utilise les méthodes Receive des objets MessageConsumer.

Mode de livraison des messages
XMS prend en charge deux modes de livraison de messages.

Distribution synchrone des messages

Les messages sont distribués de manière synchrone à une application lorsque celle-ci utilise les méthodes Receive des objets MessageConsumer.

Les méthodes Receive permettent à une application d'attendre un message indéfiniment ou pendant une période de temps spécifiée. Si une application ne veut pas attendre de message, elle peut utiliser la méthode Receive with No Wait.

Concepts associés

Sessions transactionnelles
Les applications XMS peuvent exécuter des transactions locales. Une *transaction locale* est une transaction qui implique uniquement les modifications apportées aux ressources du gestionnaire de files d'attente ou du bus d'intégration de services auquel l'application est connectée.

Accusé de réception de message
Chaque session non transactionnelle possède un mode d'accusé de réception qui détermine la façon dont l'application accuse réception des messages. Trois modes d'accusé de réception sont disponibles, et le choix du mode d'accusé de réception a un impact sur la conception de l'application.

une distribution asynchrone de messages
XMS utilise une unité d'exécution pour gérer les distributions asynchrones de messages pour une session. Cela signifie qu'une seule fonction de programme d'écoute de message ou une méthode `onMessage()` peut être exécutée à la fois.

Mode de livraison des messages
XMS prend en charge deux modes de livraison de messages.

Mode de livraison des messages

XMS prend en charge deux modes de livraison de messages.

- Les messages *persistants* sont distribués une fois. Un serveur de messagerie prend des précautions spécifiques, telles que la consignation des messages, pour s'assurer que les messages persistants ne sont pas perdus au cours du transfert, même en cas d'arrêt anormal.
- Les messages *non persistants* sont distribués une seule fois. Les messages non persistants sont moins fiables que les messages persistants car ils peuvent être perdus au cours du transfert en cas d'arrêt anormal.

Le choix du mode de livraison est un compromis entre les performances et la fiabilité. Les messages non persistants sont généralement transportés plus rapidement que les messages persistants.

Concepts associés

Sessions transactionnelles
Les applications XMS peuvent exécuter des transactions locales. Une *transaction locale* est une transaction qui implique uniquement les modifications apportées aux ressources du gestionnaire de files d'attente ou du bus d'intégration de services auquel l'application est connectée.

Accusé de réception de message
Chaque session non transactionnelle possède un mode d'accusé de réception qui détermine la façon dont l'application accuse réception des messages. Trois modes d'accusé de réception sont disponibles, et le choix du mode d'accusé de réception a un impact sur la conception de l'application.

une distribution asynchrone de messages
XMS utilise une unité d'exécution pour gérer les distributions asynchrones de messages pour une session. Cela signifie qu'une seule fonction de programme d'écoute de message ou une méthode `onMessage()` peut être exécutée à la fois.

Distribution synchrone des messages
Les messages sont distribués de manière synchrone à une application lorsque celle-ci utilise les méthodes Receive des objets MessageConsumer.

Destinations

Une application XMS utilise un objet Destination pour spécifier la destination des messages envoyés et la source des messages reçus.

Une application XMS peut créer un objet Destination au moment de l'exécution, ou obtenir une destination prédéfinie à partir du référentiel des objets gérés.

Dans une fabrique de connexions, la manière la plus souple pour une application XMS de spécifier une destination consiste à la définir en tant qu'objet géré. Ainsi, des applications écrites en langage C, C++ et .NET et des applications Java peuvent partager des définitions de destination. Les propriétés d'objets Destination gérés peuvent être changées sans modification du code.

Pour les applications .NET, vous pouvez créer une destination par l'intermédiaire de la méthode CreateTopic ou CreateQueue. Ces deux méthodes sont disponibles dans les objets ISession et XMSFactoryFactory de l'API .NET. Pour plus d'informations, voir «Destinations dans .NET», à la page 47 et «IDestination», à la page 111.

Référence associée

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

identificateur URI de rubrique

L'identificateur URI (Uniform Resource Identifier) de rubrique spécifie le nom de la rubrique ; il peut également en spécifier une ou plusieurs propriétés.

L'identificateur URI d'une rubrique commence par la séquence topic://, suivie du nom de la rubrique et, éventuellement, d'une liste de paires nom-valeur qui définissent les autres propriétés de la rubrique. Un nom de rubrique ne peut pas être vide.

Voici un exemple dans un fragment de code .NET :

```
topic = session.CreateTopic("topic://Sport/Football/Results?multicast=7");
```

Pour plus d'informations sur les propriétés d'une rubrique, y compris les noms et les valeurs admises que vous pouvez utiliser dans un identificateur URI, voir «Propriétés de Destination», à la page 189.

Lorsque vous spécifiez un identificateur URI de rubrique, vous pouvez utiliser des caractères génériques. La syntaxe pour ces caractères génériques dépend du type de connexion et de la version du courtier ; les options suivantes sont disponibles :

- Gestionnaire de files d'attente IBM WebSphere MQ 7.0 avec format générique au niveau des caractères
- Gestionnaire de files d'attente IBM WebSphere MQ 7.0 avec format générique au niveau des rubriques
- Bus d'intégration de services WebSphere Application Server

Gestionnaire de files d'attente IBM WebSphere MQ 7.0 avec format générique au niveau des caractères

Un gestionnaire de files d'attente IBM WebSphere MQ 7.0 avec format générique au niveau des caractères utilise les caractères génériques suivants :

* pour 0 caractère ou plus
? pour 1 caractère
% pour un caractère d'échappement

Le Tableau 1, à la page 30 contient quelques exemples d'utilisation de ce schéma de caractères génériques.

Tableau 1. Exemples d'URI utilisant le schéma de format générique au niveau des caractères pour le gestionnaire de files d'attente IBM WebSphere MQ 7.0

Identificateur URI	Correspondance	Exemples
"topic://Sport*Résultats"	Toutes les rubriques commençant par "Sport" et se terminant par "Résultats"	"topic://SportsRésultats" et "topic://Sport/Hockey/National/Div3/Résultats"
" topic://Sport?Résultats "	Toutes les rubriques commençant par "Sport", suivi d'un caractère unique, suivi de "Résultats"	"topic://SportsRésultats" et "topic://SportXRésultats"
"topic://Sport/*ball*/Div?/Résultats/*/???"	Rubriques	"topic://Sport/Football/Div1/Résultats/2002/Nov" et "topic://Sport/Netball/National/Div3/Résultats/02/Jan"

Gestionnaire de files d'attente IBM WebSphere MQ 7.0 avec format générique au niveau des rubriques

Un gestionnaire de files d'attente IBM WebSphere MQ 7.0 avec format générique au niveau des rubriques utilise les caractères génériques suivants :

pour correspondre à plusieurs niveaux
+ pour correspondre à un niveau unique

Le Tableau 2, à la page 30 contient quelques exemples d'utilisation de ce schéma de caractères génériques.

Tableau 2. Exemples d'URI utilisant le schéma de format générique au niveau des rubriques pour le gestionnaire de files d'attente IBM WebSphere MQ 7.0

Identificateur URI	Correspondance	Exemples
"topic://Sport/+/Résultats"	Toutes les rubriques avec un nom de niveau hiérarchique unique entre Sport et Résultats	"topic://Sport/Football/Résultats" et "topic://Sport/Ju-Jitsu/Résultats"
"topic://Sport/#/Résultats"	Toutes les rubriques commençant par "Sport" et se terminant par "/Résultats"	"topic://Sport/Football/Résultats" et "topic://Sport/Hockey/National/Div3/Résultats"
"topic://Sport/Football/#"	Toutes les rubriques commençant par "Sport/Football/"	"topic://Sport/Football/Résultats" et "topic://Sport/Football/Informations/Signatures/Gestion"

Bus d'intégration de services WebSphere Application Server

Le bus d'intégration de services WebSphere Application Server utilise les caractères génériques suivants :

* pour tous les caractères à un niveau dans la hiérarchie
// pour 0 niveau ou plus
//. pour faire correspondre 0 ou plusieurs niveaux (à la fin d'une expression Topic)

Le Tableau 3, à la page 31 contient quelques exemples d'utilisation de ce schéma de caractères génériques.

Tableau 3. Exemples d'URI utilisant le schéma de caractères génériques pour le bus d'intégration de services WebSphere Application Server

Identificateur URI	Correspondance	Exemples
"topic://Sport/*ball/Résultats"	Toutes les rubriques avec un nom de niveau hiérarchique unique se terminant par "ball" entre Sport et Résultats	"topic://Sport/Football/Résultats" et "topic://Sport/Netball/Résultats"
"topic://Sport//Résultats"	Toutes les rubriques commençant par "Sport" et se terminant par "/Résultats"	"topic://Sport/Football/Résultats" et "topic://Sport/Hockey/National/Div3/Résultats"
"topic://Sport/Football//."	Toutes les rubriques commençant par "Sport/Football/"	"topic://Sport/Football/Résultats" et "topic://Sport/Football/Informations/Signatures/Gestion"
"topic://Sport/*ball//Résultats//."	Rubriques	"topic://Sport/Football/Résultats" et "topic://Sport/Netball/National/Div3/Résultats/2002/Novembre"

Concepts associés

Identificateurs URI de file d'attente
L'identificateur URI d'une file d'attente indique le nom de cette file d'attente ; il peut également en spécifier une ou plusieurs propriétés.

Destinations temporaires
Les applications XMS peuvent créer et utiliser des destinations temporaires.

Caractères génériques de destination
XMS prend en charge les caractères génériques de destination et assure leur transmission vers l'endroit où ils sont requis pour une correspondance. Il existe un schéma de caractères génériques différent pour chaque type de serveur pouvant être associé à XMS.

Référence associée

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Identificateurs URI de file d'attente

L'identificateur URI d'une file d'attente indique le nom de cette file d'attente ; il peut également en spécifier une ou plusieurs propriétés.

L'identificateur URI d'une file d'attente commence par la séquence queue://, suivie du nom de la file d'attente ; il peut également inclure une liste de paires nom-valeur qui définissent les autre propriétés de la file d'attente.

Pour les files d'attente IBM MQ (mais pas pour les files d'attente de fournisseur de messagerie par défaut WebSphere Application Server), le gestionnaire sur lequel la file d'attente réside peut être spécifié avant la file d'attente, du moment qu'un caractère / (barre oblique) sépare le nom du gestionnaire de files d'attente et le nom de la file d'attente.

Si un gestionnaire de files d'attente est spécifié, il doit s'agir de celui auquel XMS est directement connecté pour la connexion utilisant cette file d'attente, ou il doit être accessible à partir de cette file d'attente. Les gestionnaires de files d'attente éloignées sont pris en charge uniquement pour extraire des messages des files d'attente, et non pour placer des messages dans des files d'attente. Pour plus de détails, reportez-vous à la documentation du gestionnaire de files d'attente IBM MQ.

Si aucun gestionnaire de files d'attente n'est spécifié, le séparateur / (barre oblique) supplémentaire est facultatif ; sa présence ou son absence ne fait aucune différence pour la définition de la file d'attente.

Les définitions de file d'attente suivantes sont toutes équivalentes pour une file d'attente IBM MQ appelée QB sur un gestionnaire de files d'attente appelé QM_A, auquel XMS est directement connecté:

```
queue://QB
queue:///QB
queue://QM_A/QB
```

Concepts associés

identificateur URI de rubrique
L'identificateur URI (Uniform Resource Identifier) de rubrique spécifie le nom de la rubrique ; il peut également en spécifier une ou plusieurs propriétés.

Destinations temporaires
Les applications XMS peuvent créer et utiliser des destinations temporaires.

Caractères génériques de destination
XMS prend en charge les caractères génériques de destination et assure leur transmission vers l'endroit où ils sont requis pour une correspondance. Il existe un schéma de caractères génériques différent pour chaque type de serveur pouvant être associé à XMS.

Référence associée

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Destinations temporaires

Les applications XMS peuvent créer et utiliser des destinations temporaires.

Généralement, une application utilise une destination temporaire pour recevoir des réponses à des messages de demande. Pour spécifier la destination où la réponse à un message de demande doit être envoyé, une application appelle la méthode Set JMSReplyTo de l'objet Message représentant le message de demande. La destination spécifiée sur l'appel peut être une destination temporaire.

Bien qu'une session soit utilisée pour créer une destination temporaire, la portée d'une destination temporaire est en fait la connexion qui a été utilisée pour créer la session. Toutes les sessions de la connexion peuvent créer des expéditeurs de message et des consommateurs de message pour la destination temporaire. La destination temporaire reste active jusqu'à sa suppression explicite ou la fin de la connexion.

Lorsqu'une application crée une file d'attente temporaire, une file d'attente est créé dans le serveur de messagerie auquel l'application est connectée. Si l'application est connectée à un gestionnaire de files d'attente, une file d'attente dynamique est créée à partir de la file d'attente modèle dont le nom est spécifié par la propriétéXMSC_WMQ_TEMPORARY_MODEL ; le préfixe utilisé pour former le nom de la file d'attente dynamique est spécifié par la propriété XMSC_WMQ_TEMP_Q_PREFIX. Si l'application est connectée à un bus d'intégration de services, une file d'attente temporaire est créée dans le bus, et le préfixe qui est utilisé pour former le nom de la file d'attente temporaire est spécifié par la propriété XMSC_WPM_TEMP_Q_PREFIX.

Lorsqu'une application connectée à un bus d'intégration de services crée une rubrique temporaire, le préfixe qui est utilisé pour former le nom de la rubrique est spécifié par la propriété XMSC_WPM_TEMP_TOPIC_PREFIX.

Concepts associés

identificateur URI de rubrique
L'identificateur URI (Uniform Resource Identifier) de rubrique spécifie le nom de la rubrique ; il peut également en spécifier une ou plusieurs propriétés.

Identificateurs URI de file d’attente
L'identificateur URI d'une file d'attente indique le nom de cette file d'attente ; il peut également en spécifier une ou plusieurs propriétés.

Caractères génériques de destination
XMS prend en charge les caractères génériques de destination et assure leur transmission vers l'endroit où ils sont requis pour une correspondance. Il existe un schéma de caractères génériques différent pour chaque type de serveur pouvant être associé à XMS.

Référence associée

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Caractères génériques de destination

XMS prend en charge les caractères génériques de destination et assure leur transmission vers l'endroit où ils sont requis pour une correspondance. Il existe un schéma de caractères génériques différent pour chaque type de serveur pouvant être associé à XMS.

Les schémas sont les suivants :

Type de connexion	Schéma de caractères génériques	Description
WebSphere MQ gestionnaire de files d'attente	*	0 caractère ou plus
	?	1 caractère
	%	Caractère d'échappement
Connexion en temps réel à un courtier	#	Correspondance à plusieurs niveaux
	+	Correspondance à un niveau unique
bus d'intégration de services WebSphere	*	Correspondance à tous les caractères d'un niveau dans la hiérarchie
	//	Correspondance à 0 niveau ou plus
	//.	Correspondance à 0 niveau ou plus (à la fin d'une expression Topic)

Concepts associés

identificateur URI de rubrique
L'identificateur URI (Uniform Resource Identifier) de rubrique spécifie le nom de la rubrique ; il peut également en spécifier une ou plusieurs propriétés.

Identificateurs URI de file d’attente
L'identificateur URI d'une file d'attente indique le nom de cette file d'attente ; il peut également en spécifier une ou plusieurs propriétés.

Destinations temporaires
Les applications XMS peuvent créer et utiliser des destinations temporaires.

Référence associée

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Expéditeurs de message

Dans XMS, un expéditeur de message peut être créé avec ou sans destination associée. Si l'expéditeur de message est créé avec une destination null, une destination valide doit être spécifiée à l'envoi d'un message.

Expéditeurs de messages sans destination associée

Dans XMS .NET, un expéditeur de message peut être créé avec une destination Null.

Pour créer un expéditeur de message sans destination associée lors de l'utilisation de l'API .NET, la valeur NULL doit être transmise en tant que paramètre dans la méthode `CreateProducer()` de l'objet ISession (par exemple, `session.CreateProducer(null)`). Toutefois, une destination valide doit être spécifiée lors de l'envoi du message.

Expéditeurs de messages avec destination associée

Dans ce scénario, l'expéditeur de message est créé avec une destination valide. Au cours de l'opération d'envoi, la destination ne doit pas être indiquée.

Consommateurs de message

Les consommateurs de message peuvent être classés comme consommateurs durables ou non durables, et comme consommateurs de message synchrone ou asynchrone.

Abonnés durables

Un abonné durable est un consommateur de message qui reçoit tous les messages publiés sur une rubrique, y compris les messages publiés pendant qu'il est inactif.

Les informations de cette rubrique sont pertinentes uniquement si une application se connecte à un gestionnaire de files d'attente IBM MQ ou à un bus d'intégration de services WebSphere Application Server. Elles ne le sont pas pour une connexion en temps réel à un courtier.

Pour créer un abonné durable à une rubrique, une application appelle la méthode Create Durable Subscriber d'un objet Session, en spécifiant sous forme de paramètres un nom identifiant l'abonnement durable et un objet Destination représentant la rubrique. L'application peut créer un abonné durable avec ou sans sélecteur de message ; elle peut également spécifier si l'abonné durable peut recevoir des messages publiés par sa propre connexion.

La session utilisée pour créer un abonné durable doit avoir un identificateur de client associé. L'identificateur de client est le même que celui associé à la connexion utilisée pour créer la session ; il est spécifié selon la méthode décrite dans «Objets ConnectionFactories et Connection», à la page 21.

Le nom qui identifie l'abonnement durable doit être unique dans l'identificateur de client ; cet identificateur fait donc partie de l'identificateur unique de l'abonnement durable. Le serveur de messagerie conserve un enregistrement de l'abonnement durable et garantit que tous les messages publiés sur la rubrique sont conservés jusqu'à l'envoi d'un accusé de réception de la part de l'abonné durable ou jusqu'à leur expiration.

Le serveur de messagerie gère l'enregistrement de l'abonnement durable même après la fermeture de l'abonné durable. Pour réutiliser un abonnement durable précédemment créé, une application doit créer un abonné durable en spécifiant le même nom d'abonnement et en utilisant une session avec le même identificateur de client que ceux utilisés pour cet abonnement durable. Une seule session à la fois peut disposer d'un abonné durable pour un abonnement durable particulier.

La portée d'un abonnement durable est le serveur de messagerie qui gère un enregistrement de l'abonnement. Si deux applications connectées chacun à un serveur de messagerie distinct créent un abonné durable avec le même nom d'abonnement et le même identificateur de client, deux abonnements durables indépendants sont créés.

Pour supprimer un abonnement durable, une application appelle la méthode Unsubscribe d'un objet Session, en spécifiant sous forme de paramètre le nom qui identifie cet abonnement durable. L'identificateur de client associé à la session doit être le même que celui qui est associé à l'abonnement durable. Le serveur de messagerie supprime l'enregistrement de l'abonnement durable qu'il gère et n'envoie plus de messages à l'abonné durable.

Pour modifier un abonnement existant, une application peut créer un abonné durable avec le même nom d'abonnement et le même identificateur de client, mais en spécifiant une rubrique et/ou un sélecteur de message différents. Modifier un abonnement durable équivaut à supprimer cet abonnement et à en créer un nouveau.

Pour une application qui se connecte au gestionnaire de files d'attente IBM WebSphere MQ 7.0, XMS gère les files d'attente de souscription. L'application ne doit donc pas nécessairement spécifier une file d'attente d'abonné. XMS ignore la file d'attente d'abonné lorsque celle-ci est spécifiée.

Toutefois, pour une application qui se connecte au gestionnaire de files d'attente IBM WebSphere MQ 6.0, une file d'attente de souscription doit être désignée pour chaque abonné durable. Pour spécifier le nom de la file d'attente d'abonné pour une rubrique, définissez la propriété XMSC_WMQ_DUR_SUBQ de l'objet Destination représentant cet objet. La file d'attente d'abonné par défaut est SYSTEM.JMS.D.SUBSCRIBER.QUEUE.

Les abonnés durables se connectant aux gestionnaires de file d'attente IBM WebSphere MQ 6.0 peuvent partager une file d'attente de souscription unique, ou chaque abonné peut extraire ses messages à partir de sa propre file d'attente de souscription exclusive. Pour plus d'informations sur l'approche à adopter pour votre application, voir *XMSIBM WebSphere MQ Utilisation de Java*.

Notez que vous ne pouvez pas modifier la file d'attente d'abonnement pour un abonnement durable. La seule manière de procéder consiste à supprimer l'abonnement et à en créer un nouveau.

Pour une application qui se connecte à un bus d'intégration de services, chaque abonné durable doit avoir un accueil d'abonnement durable. Pour spécifier cet accueil pour tous les abonnés durables utilisant la même connexion, définissez la propriété XMSC_WPM_DUR_SUB_HOME de l'objet ConnectionFactory utilisé pour créer la connexion. Pour spécifier cet accueil pour une rubrique particulière, définissez la propriété XMSC_WPM_DUR_SUB_HOME de l'objet Destination représentant la rubrique. Un accueil d'abonnement durable doit être spécifié pour une connexion avant qu'une application puisse créer un abonné durable qui utilise la connexion. Toute valeur spécifiée pour une destination remplace la valeur spécifiée pour la connexion.

Abonnés non durables

Un abonné non durable est un consommateur de message qui reçoit uniquement des messages publiés lorsqu'il est actif. Les messages livrés lorsque l'abonné est inactif sont perdus.

Les informations de cette rubrique sont pertinentes uniquement lorsque vous utilisez la messagerie de publication/abonnement sur un gestionnaire de files d'attente IBM WebSphere MQ 6.0.

Si les objets consommateur ne sont pas supprimés avant ou pendant la fermeture de la connexion, les messages peuvent rester sur les files d'attente du courtier pour les abonnés qui ne sont plus actifs.

Dans ce cas, les messages de ces files d'attente peuvent être supprimés via l'utilitaire Cleanup, fourni avecIBM WebSphere MQ classes for JMS Classes for JMS. Vous trouverez des informations sur cet utilitaire dans *IBM WebSphere MQ Using Java.* Vous pouvez également augmenter le nombre de lignes de la file d'attente de l'abonné si cette dernière contient encore beaucoup de messages.

Consommateurs de messages synchrones

Le consommateur de messages synchrones reçoit les messages en provenance d'une file d'attente de manière synchrone.

Un consommateur de messages synchrones reçoit un message à la fois. Lorsque la méthode `Receive(wait interval)` est utilisée, l'appel attend le message pendant une période de temps spécifiée (en millisecondes) ou jusqu'à la fermeture du consommateur de message.

Lorsque la méthode ReceiveNoWait() est utilisée, le consommateur de messages synchrones reçoit les messages sans délai ; si le message suivant est disponible, il est reçu immédiatement, sinon, un pointeur vers un objet Message Null est renvoyé.

Consommateurs de message asynchrone

Le consommateur de message asynchrone reçoit un message en provenance d'une file d'attente de manière asynchrone. Le programme d'écoute de message enregistré par l'application est appelé chaque fois qu'un nouveau message est disponible dans la file d'attente.

Messages incohérents dans XMS

Un message incohérent est un message qui ne peut pas être traité par une application MDB. Si un message incohérent est détecté, l'objet XMS MessageConsumer peut le remettre en file d'attente en fonction de deux propriétés de file d'attente, BOQUEUE et BOTHRESH.

Dans certaines circonstances, un message distribué à une application MDB peut être remis dans une file d'attente IBM MQ. Cela peut se produire, par exemple, si un message est distribué au sein d'une unité de travail qui est ensuite annulée. Un message annulé est généralement redistribué, mais un message formaté de manière incorrecte peut provoquer des échecs répétés de l'application MDB et donc ne pas être distribué. Un message de ce type est appelé un message incohérent. Vous pouvez configurer IBM MQ de sorte que le message incohérent soit automatiquement transféré à une autre file d'attente pour investigation ou supprimé. Pour plus d'informations sur une telle configuration de IBM MQ, voir Gestion des messages incohérents dans ASF.

Parfois, un message formaté de manière incorrecte est placé dans une file d'attente. Dans ce contexte, cela signifie que l'application de réception ne peut pas traiter correctement le message. Un tel message peut provoquer l'échec de l'application de réception et l'annulation du message formaté de manière incorrecte. Le message peut ensuite être distribué plusieurs fois sur la file d'entrée et plusieurs fois refusé par l'application. Ces messages sont appelés des messages incohérents. L'objet XMS MessageConsumer détecte les messages incohérents et les réachemine vers une autre destination.

Le gestionnaire de files d'attente IBM MQ garde un enregistrement du nombre de fois où chaque message a été annulé. Lorsque ce nombre atteint une valeur de seuil configurable, le consommateur de message replace le message dans une file d'attente d'annulation nommée. Si cette opération échoue, le message est supprimé de la file d'attente d'entrée, puis placé dans une file d'attente de messages non livrés ou détruit.

Les objets XMS ConnectionConsumer gèrent les messages incohérents de la même manière et avec les mêmes propriétés de file d'attente. Si plusieurs consommateurs de connexion surveillent une même file d'attente, il est possible que le message incohérent soit distribué à une application un nombre de fois supérieur à la valeur de seuil avant la remise en file d'attente. Ce comportement s'explique par la manière dont les clients de connexion individuelle gèrent les files d'attente et remettre en file d'attente les messages incohérents.

La valeur de seuil et le nom de la file d'attente d'annulation sont des attributs d'une file d'attente IBM MQ . Les noms des attributs sont BackoutThreshold et BackoutRequeueQName. La file d'attente à laquelle ils s'appliquent est la suivante :

- Pour la messagerie point-à-point, il s'agit de la file d'attente locale sous-jacente. Ceci est important lorsque les consommateurs de message et les clients de connexion utilisent des alias de file d'attente.
- Pour la messagerie publication/abonnement en mode normal de fournisseur de messagerie IBM MQ, il s'agit de la file d'attente modèle à partir de laquelle la file d'attente gérée de Topic est créée.
- Pour la messagerie publication/abonnement en mode de migration de fournisseur de messagerie IBM MQ, il s'agit de la file d'attente CCSUB définie sur l'objet TopicConnectionFactory, ou de la file d'attente CCDSUB définie sur l'objet Topic.

Pour définir les attributs BackoutThreshold et BackoutRequeueQName, entrez le commandes MQSC suivantes :

```
ALTER QLOCAL(your.queue.name) BOTHRESH(threshold value)
BOQUEUE(your.backout.queue.name)
```

Pour la messagerie publication/abonnement, si votre système crée une file d'attente dynamique pour chaque abonnement, ces valeurs d'attribut sont obtenues à partir de la file d'attente modèle WebSphere MQ classes for JMS SYSTEM.JMS.MODEL.QUEUE. Pour modifier ces paramètres, utilisez :

```
ALTER QMODEL(SYSTEM.JMS.MODEL.QUEUE) BOTHRESH(threshold value)
BOQUEUE(your.backout.queue.name)
```

Si la valeur de seuil d'annulation est zéro, la gestion des messages incohérents est désactivée et ces messages restent en file d'attente d'entrée. Dans le cas contraire, lorsque le nombre d'annulations atteint la valeur de seuil, le message est envoyé à la file d'attente d'annulation nommée.

Si le nombre d'annulations atteint la valeur de seuil, mais que le message ne peut pas être placé dans la file d'attente d'annulation, il est envoyé à la file d'attente de messages non livrés, ou supprimé s'il s'agit d'un message non persistant.

Cette situation se produit si la file d'attente d'annulation n'est pas définie ou si l'objet MessageConsumer ne peut pas envoyer le message à la file d'attente d'annulation.

Configuration de votre système pour le traitement des messages incohérents

La file d'attente utilisée par XMS .NET lorsqu'il interroge les attributs **BOTHRESH** et **BOQNAME** dépend du style de messagerie :

- Pour la messagerie point-à-point, il s'agit de la file d'attente locale sous-jacente. Elle est importante lorsqu'une application XMS .NET consomme des messages provenant de files d'attente alias ou de files d'attente de cluster.
- Pour la messagerie de publication/abonnement, une file d'attente gérée est créée afin de contenir les messages pour une application. XMS .NET interroge la file d'attente gérée afin de déterminer les valeurs pour les attributs **BOTHRESH** et **BOQNAME**.

  La file d'attente gérée est créée à partir d'une file d'attente modèle associée à l'objet Topic auquel l'application s'est abonnée et hérite des valeurs des attributs **BOTHRESH** et **BOQNAME** spécifiés dans la file d'attente modèle. La file d'attente modèle qui est utilisée varie selon que l'application de réception a souscrit un abonnement durable ou non durable :

  – La file d'attente modèle utilisée pour les abonnements durables est spécifiée par l'attribut **MDURMDL** de l'objet Topic. La valeur par défaut de cet attribut est SYSTEM.DURABLE.MODEL.QUEUE.

  – Pour les abonnements non durables, la file d'attente modèle utilisée est spécifiée par l'attribut **MNDURMDL**. La valeur par défaut de l'attribut **MNDURMDL** est SYSTEM.NDURABLE.MODEL.QUEUE.

Lors de l'interrogation des attributs **BOTHRESH** et **BOQNAME**, XMS .NET :

- Ouvre la file d'attente locale, ou la file d'attente cible pour une file d'attente d'alias.
- Interroge les attributs **BOTHRESH** et **BOQNAME**.
- Ferme la file d'attente locale, ou la file d'attente cible pour une file d'attente alias.

Les options d'ouverture utilisées lors de l'ouverture d'une file d'attente locale, ou d'une file d'attente cible pour une file d'attente alias, dépendent de la version d'IBM MQ qui est utilisée :

- Pour la IBM MQ 9.0.0 Fix Pack 8 et les versions antérieures, si la file d'attente locale, ou la file d'attente cible pour une file d'attente alias, est une file d'attente de cluster, XMS .NET ouvre la file d'attente avec les options MQOO_INPUT_AS_Q_DEF, MQOO_INQUIRE et MQOO_FAIL_IF_QUIESCING. Cela signifie que l'utilisateur qui exécute l'application de réception doit disposer de l'accès en interrogation et en obtention dans l'instance locale de la file d'attente de cluster.

  XMS .NET ouvre tous les autres types de file d'attente locale avec les options d'ouverture MQOO_INQUIRE et MQOO_FAIL_IF_QUIESCING. Pour que XMS .NET interroge les valeurs des attributs, l'utilisateur qui exécute l'application de réception doit disposer de l'accès en interrogation dans la file d'attente locale.

- V 9.0.0.9 Dans le cadre de l'utilisation de XMS .NET depuis la IBM MQ 9.0.0 Fix Pack 9, l'utilisateur qui exécute l'application de réception doit disposer de l'accès en interrogation dans la file d'attente locale, quel que soit le type de la file d'attente.

Pour déplacer les messages incohérents dans une file de remise en attente suite à une annulation ou dans la file d'attente de rebut du gestionnaire de files d'attente, vous devez accorder à l'utilisateur exécutant l'application les droits `put` et `passall`.

Gestion des messages incohérents dans ASF
Lorsque vous utilisez ASF (Application Server Facilities), les messages incohérents sont traités par l'objet ConnectionConsumer plutôt que par l'objet MessageConsumer. ConnectionConsumer replace les messages en file d'attente en fonction des propriétés BackoutThreshold et BackoutRequeueQName de celle-ci.

Lorsqu'une application utilise ConnectionConsumers, les circonstances dans lesquelles un message est annulé dépend de la session fournie par le serveur d'application :

- Dans le cas d'une session non transactionnelle avec AUTO_ACKNOWLEDGE ou DUPS_OK_ACKNOWLEDGE, un message est annulé uniquement après une erreur système ou un arrêt imprévu de l'application.
- Dans le cas d'une session non transactionnelle avec CLIENT_ACKNOWLEDGE, les messages pour lesquels aucun accusé de réception n'a été envoyé peuvent être annulés par le serveur d'application via un appel `Session.recover()`.

  Généralement, l'implémentation client de MessageListener ou le serveur d'application appelle `Message.acknowledge()`. `Message.acknowledge()` accuse réception de tous les messages distribués jusqu'à présent au cours de la session.
- Dans le cas d'une session transactionnelle, les messages pour lesquels aucun accusé de réception n'a été envoyé peuvent être annulés par le serveur d'application via un appel `Session.rollback()`.

Navigateurs de file d'attente

Une application utilise un navigateur de files d'attente pour parcourir les messages d'une file d'attente sans les supprimer.

Pour créer un navigateur de files d'attente, une application appelle la méthode Create Queue Browser de l'objet ISession, en spécifiant en tant que paramètre un objet Destination qui identifie la file d'attente à parcourir. L'application peut créer un navigateur de files d'attente avec ou sans sélecteur de message.

Après avoir créé un navigateur de files d'attente, l'application peut appeler la méthode GetEnumerator de l'objet IQueueBrowser pour obtenir la liste des messages en file d'attente. La méthode renvoie une expression énumérative qui encapsule une liste d'objets Message. L'ordre des objets Message dans la liste est le même que celui dans lequel les messages seront extraits de la file d'attente. L'application peut alors utiliser l'expression énumérative pour parcourir chaque message à son tour.

L'expression énumérative est mise à jour de manière dynamique à mesure que les messages sont insérés dans la file d'attente et supprimés de la file d'attente. Chaque fois que l'application appelle IEnumerator.MoveNext () pour parcourir le message suivant dans la file d'attente, le message reflète le contenu en cours de la file d'attente.

Une application peut appeler la méthode GetEnumerator plusieurs fois pour un navigateur de files d'attente donné. Chaque appel renvoie une nouvelle expression énumérative. L'application peut ainsi utiliser plus d'une expression énumérative pour parcourir les messages dans une file d'attente et gérer plusieurs positions dans la file d'attente.

Une application peut utiliser un navigateur de files d'attente pour rechercher un message à supprimer d'une file d'attente, puis faire appel à un consommateur de message avec sélecteur pour supprimer le message. Le sélecteur peut choisir le message en fonction de la valeur de la zone d'en-tête JMSMessageID. Pour plus d'informations sur les zones d'en-tête de message JMS, voir «Zones d'en-tête dans un message XMS», à la page 71.

Demandeurs

Une application utilise un demandeur pour envoyer un message de demande et ensuite, attendre et recevoir la réponse.

De nombreuses applications de messagerie sont basés sur des algorithmes qui envoient un message de demande et attendent ensuite une réponse. XMS fournit une classe appelée Requestor pour vous aider au développement de ce type d'application.

Pour créer un demandeur, une application appelle le constructeur Create Requestor de la classe Requestor et spécifie en tant que paramètres un objet Session et un objet Destination qui identifie l'emplacement où les messages de demande devront être envoyés. La session ne doit pas être transactionnelle, ni être associée au mode d'accusé de réception XMSC_CLIENT_ACKNOWLEDGE. Le constructeur crée automatiquement une file d'attente ou une rubrique temporaire où les messages de réponse devront être envoyés.

Après avoir créé un demandeur, l'application peut appeler la méthode Request de l'objet Requestor pour envoyer un message de demande, puis attendre et recevoir une réponse de l'application à laquelle le message de demande a été envoyé. L'appel attend la réponse jusqu'à sa réception ou jusqu'à ce que la session prenne fin. Une seule réponse est requise par le demandeur pour chaque message de demande.

Lorsque l'application ferme le demandeur, la file d'attente ou la rubrique temporaire est supprimée. Toutefois, la session associée n'est pas fermée.

Suppression d'objet

Lorsqu'une application supprime un objet XMS qu'elle a créé, XMS libère les ressources internes qui ont été allouées à cet objet.

Lorsqu'une application crée un objet XMS, XMS alloue de la mémoire et d'autres ressources internes à cet objet. XMS réserve ces ressource internes jusqu'à ce que l'application supprime l'objet par l'intermédiaire d'une méthode de fermeture et de suppression ; XMS libère alors les ressources internes. Si une application tente de supprimer un objet qui l'a déjà été, l'appel est ignoré.

Lorsqu'une application supprime un objet Connection ou Session, XMS supprime automatiquement certains objets associés et libère leurs ressources internes. Sont concernés les objets créés par l'objet Connection ou Session et n'ayant aucune fonction indépendante. Ces objets sont présentés dans Tableau 4, à la page 39.

Remarque : Si une application ferme une connexion avec des sessions dépendantes, tous les objets dépendant de ces sessions sont également supprimés. Seuls les objets Connection et Session peuvent avoir des objets dépendants.

Tableau 4. Objets supprimés automatiquement

Objet supprimé	Méthode	Objets dépendants supprimés automatiquement
Connexion	Close Connection	Objets ConnectionMetaData et Session
Session	Close Session	Objets MessageConsumer, MessageProducer, QueueBrowser et Requestor

Transactions XA IBM MQ gérées via XMS

Les transactions XA IBM MQ gérées peuvent être utilisées via XMS.

Pour utiliser les transactions XA via XMS, une session transactionnelle doit être créée. Lorsqu'une transaction XA est en cours d'utilisation, la commande de transaction se fait via les transactions DTC (Distributed Transaction Coordinator) globales et non via les sessions XMS. Lors de l'utilisation de transactions XA, `Session.commit` ou `Session.rollback` ne peut pas être exécutée sur la session XMS. A la place, utilisez la méthode DTC `Transscope.Commit` ou `Transscope.Rollback` pour valider ou annuler les transactions. Si une session est utilisée pour une transaction XA, le fournisseur ou le consommateur créé à l'aide de la session doit faire partie de la transaction XA. Ils ne peuvent pas

être utilisés pour des opérations externes à la transaction XA. Ils ne peuvent pas être utilisés pour des opérations telles que `Producer.send` or `Consumer.receive` en dehors de la transaction XA.

Un objet exception `IllegalStateException` est émis si

- Une session transactionnelle XA est utilisée pour `Session.commit` ou `Session.rollback`.
- Les objets fournisseurs ou consommateurs utilisés une fois dans une session transactionnelle XA sont utilisés hors de la portée de la transaction XA.

Les transactions XA ne sont pas prises en charge dans les consommateurs asynchrones.

Remarque :

1. Une fermeture peut se produire sur l'objet `Producer`, `Consumer`, `Session` ou `Connection` avant la validation de la transaction XA. Dans ce cas, les messages de la transaction sont annulés. De même, si la connexion est interrompue avant la validation de la transaction XA, tous les messages de la transaction sont annulés. Pour un objet `Producer`, une annulation signifie que les messages ne sont pas placés en file d'attente. Pour un objet `Consumer`, une annulation signifie que les messages restent en file d'attente.
2. Si un objet `Producer` place un message avec `TimeToLive` dans `TransactionScope` et qu'un `commit` est émis une fois que le temps est écoulé, le message peut expirer avant que le `commit` soit émis. Dans ce cas, le message n'est pas disponible pour les objets `Consumer`.
3. Les objets `Session` ne sont pas pris en charge entre unités d'exécution. L'utilisation de transactions avec des objets `Session` partagés entre unités d'exécution n'est pas prise en charge.

Types primitifs XMS

XMS fournit des équivalents pour les huit types primitifs Java (byte, short, int, long, float, double, char et boolean). Ils permettent l'échange de messages entre XMS et JMS sans perte ni corruption de données.

Le Tableau 5, à la page 40 répertorie l'équivalent Java du type de données, de la taille et de la valeur minimale et maximale pour chaque type de primitif XMS.

Tableau 5. Types de données XMS et leur équivalent Java

Type de données XMS	Type de données Java compatible	Taille	Valeur minimale	Valeur maximale
System.Boolean	boolean	32 bits	false	conforme
System.SBYTE	byte	8 bits	-2^7 (-128)	2^7-1 (127)
System.BYTE	byte	8 bits	-2^7 (-128)	2^7-1 (127)
System.CHAR	byte	8 bits	-2^7 (-128)	2^7-1 (127)
System.Int16	court	16 bits	-2^{15} (-32768)	2^{15}-1 (32767)
System.Int32	int	32 bits	-2^{31} (-2147483648)	2^{31}-1 (2147483647)
System.Int64	long	64 bits	-2^{63} (-9223372036854775808)	2^{63}-1 (9223372036854775807)
System.Single	float	32 bits	-3.402823E-38 (précision à 7 chiffres après la virgule)	3.402823E+38 (précision à 7 chiffres après la virgule)
System.Double	double	64 bits	-1.79769313486231E-308 (précision à 15 chiffres après la virgule)	1.79769313486231E+308 (précision à 15 chiffres après la virgule)

Concepts associés

Attributs et propriétés des objets

Un objet XMS peut être caractérisé par des attributs et des propriétés, implémentés de différentes manières.

Conversion implicite d'une valeur de propriété d'un type de données à un autre
Lorsqu'une application extrait la valeur d'une propriété, celle-ci peut être convertie par XMS dans un autre type de données. De nombreuses règles régissent les conversions prises en charge et la manière dont XMS les réalise.

Référence associée

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Conversion implicite d'une valeur de propriété d'un type de données à un autre

Lorsqu'une application extrait la valeur d'une propriété, celle-ci peut être convertie par XMS dans un autre type de données. De nombreuses règles régissent les conversions prises en charge et la manière dont XMS les réalise.

Une propriété d'objet a un nom et une valeur ; un type de données est associé à la valeur, et cette valeur est également appelée *type de propriété.*

Une application utilise les méthodes de la classe PropertyContext pour extraire et définir les propriétés des objets. Pour extraire la valeur d'une propriété, une application appelle la méthode appropriée au type de propriété. Par exemple, pour extraire la valeur d'une propriété de type entier, une application appelle la méthode GetIntProperty.

Toutefois, lorsqu'une application extrait la valeur d'une propriété, cette valeur peut être convertie par XMS dans un type de données différent. Par exemple, pour extraire la valeur d'une propriété de type entier, une application peut appeler la méthode GetStringProperty, qui renvoie cette valeur sous forme de chaîne. Les conversions prises en charge par XMS sont présentées dans Tableau 6, à la page 41.

Tableau 6. Conversions prises en charge d'un type de propriété vers d'autres types de données

Type de propri□t□	**Types de données cible pris en charge**
System.String	System.Boolean, System.Double, System.Float, System.Int32, System.Int64, System.SByte, System.Int16
System.Boolean	System.String, System.SByte, System.Int32, System.Int64, System.Int16
System.Char	System.String
System.Double	System.String
System.Float	System.String, System.Double
System.Int32	System.String, System.Int64
System.Int64	System.String
System.SByte	System.String, System.Int32, System.Int64, System.Int16
System.SByte array	System.String
System.Int16	System.String, System.Int32, System.Int64

Les règles générales suivantes régissent les conversions prises en charge :

- Les valeurs de propriétés numériques peuvent être converties d'un type de données à un autre, du moment qu'aucune donnée n'est perdue au cours de la conversion. Par exemple, la valeur d'une propriété de type System.Int32 peut être convertie en une valeur de type System.Int64, mais pas en une valeur de type System.Int16.

- Une valeur de propriété de tout type de données peut être convertie en chaîne.
- Une valeur de propriété de type chaîne peut être convertie en tout autre type de données, du moment que la chaîne est correctement formatée pour la conversion. Si une application tente de convertir une valeur de propriété de type chaîne qui n'est pas formatée correctement, XMS risque de retourner des erreurs.
- Si une application tente d'effectuer une conversion qui n'est pas prise en charge, XMS risque de retourner une erreur.

Les règles suivantes s'appliquent lorsqu'une valeur de propriété est convertie d'un type de données à un autre :

- Lors de la conversion d'une valeur de propriété de type booléen en type chaîne, la valeur true est convertie en une chaîne "true", et la valeur false est convertie en une chaîne "false".
- Lors de la conversion d'une valeur de propriété de type booléen en type numérique, y compris System.SByte, la valeur true est convertie en 1, et la valeur false est convertie en 0.
- Lors de la conversion d'une valeur de propriété de type chaîne en type booléen, la chaîne "true" (non sensible à la casse) ou "1" et convertie en true, et la chaîne "false" (non sensible à la casse) ou "0" est convertie en false. Toutes les autres chaînes ne peut pas être converties.
- Lors de la conversion d'une valeur de propriété de type chaîne en une valeur de type System.Int32, System.Int64, System.SByte ou System.Int16, la chaîne doit avoir le format suivant :

  [*blancs*][*signe*]*chiffres*

  Les composants de la chaîne sont définis comme suit :

  blancs
  Caractères blancs facultatifs de début.

  signe
  Caractère facultatif de signe plus (+) ou moins (-).

  chiffres
  Séquence contiguë de caractères numériques (0-9). Au moins un chiffre doit être présent.

  Après la séquence de caractères numériques, la chaîne peut contenir d'autres caractères autres que des chiffres, mais la conversion s'arrête au premier caractère de ce type. La chaîne est supposée représenter un entier décimal.

  XMS peut renvoyer une erreur si la chaîne n'est pas formatée correctement.
- Lors de la conversion d'une valeur de propriété de type chaîne en une valeur de type System.Double ou System.Float, la chaîne doit avoir le format suivant :

  [*blancs*][*signe*][*chiffres*][*point*[*d_chiffres*]][*e_car*[*e_signe*]*e_chiffres*]

  Les composants de la chaîne sont définis comme suit :

  blancs
  Caractères blancs facultatifs de début.

  signe
  Caractère facultatif de signe plus (+) ou moins (-).

  chiffres
  Séquence contiguë de caractères numériques (0-9). Au moins un chiffre doit être présent dans *chiffres* ou *d_chiffres*.

  point
  (Facultatif) Point décimal (.).

  d_chiffres
  Séquence contiguë de caractères numériques (0-9). Au moins un chiffre doit être présent dans *chiffres* ou *d_chiffres*.

  e_car
  Caractère exposant, qui peut être *E* ou *e*.

e_signe
Caractère facultatif de signe plus (+) ou moins (-) pour l'exposant.

e_chiffres
Séquence contiguë de caractères numériques (0-9) pour l'exposant. Au moins un chiffre doit être présent si la chaîne contient un caractère exposant.

Après la séquence de chiffres, la chaîne peut contenir d'autres caractères autres que des chiffres, mais la conversion s'arrête au premier caractère de ce type. La chaîne est supposée représenter un nombre décimal en virgule flottante avec un exposant puissance de 10.

XMS peut renvoyer une erreur si la chaîne n'est pas formatée correctement.

- Lors de la conversion d'une valeur de propriété numérique en une chaîne, y compris une valeur de propriété de type System.SByte, la valeur est convertie en une représentation de chaîne sous forme de nombre décimal et non sous forme de chaîne de caractères ASCII pour cette valeur. Par exemple, l'entier 65 est converti en chaîne "65", et non en chaîne "A".
- Lors de la conversion d'une valeur de propriété de type tableau d'octets en une chaîne, chaque octet est converti dans les 2 caractères hexadécimaux représentant cet octet. Par exemple, le tableau d'octets {0xF1, 0x12, 0x00, 0xFF} est converti en chaîne "F11200FF".

Les conversions d'un type de propriété en d'autres types de données sont prises en charge par les méthodes des classes Property et PropertyContext.

Concepts associés

Attributs et propriétés des objets
Un objet XMS peut être caractérisé par des attributs et des propriétés, implémentés de différentes manières.

Types primitifs XMS
XMS fournit des équivalents pour les huit types primitifs Java (byte, short, int, long, float, double, char et boolean). Ils permettent l'échange de messages entre XMS et JMS sans perte ni corruption de données.

Référence associée

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Messages de flux
Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Éléments itératifs

Un itérateur encapsule une liste d'objets et un curseur qui gère la position en cours dans la liste. Le concept d'un itérateur tel que disponible dans IBM Message Service Client for C/C++ est implémenté par l'utilisation de l'interface IEnumerator dans IBM Message Service Client for .NET.

Lorsqu'un itérateur est créé, le curseur est positionné avant le premier objet. Une application utilise un itérateur pour extraire chaque objet à son tour.

La classe Iterator d'IBM Message Service Client for C/C++ est équivalente à la classe Enumerator de Java.XMS .NET est similaire à Java et utilise une interface IEnumerator.

Une application peut utiliser une interface IEnumerator pour effectuer les tâches suivantes :

- Extraire les propriétés d'un message
- Extraire les paires nom-valeur du corps d'un message de mappe
- Consulter les messages dans une file d'attente

- Extraire les nom des propriétés de message définies par JMS prises en charge par une connexion

Identificateurs de jeu de caractères codés

Dans XMS .NET, toutes les chaînes sont transmises à l'aide de la chaîne .NET native. Celle-ci utilisant un codage fixe, aucune information supplémentaire n'est requise pour son interprétation. Par conséquent, la propriété XMSC_CLIENT_CCSID n'est pas requise pour les applications XMS .NET.

Codes d'erreur et d'exception XMS

XMS utilise une plage de codes d'erreur pour signaler les pannes. Ces codes d'erreur ne sont pas listés explicitement dans cette documentation, car ils peuvent varier d'une édition à une autre. Seuls les codes d'exception XMS (au format XMS_X_...) sont documentés car ils restent identiques quelle que soit l'édition de XMS.

Génération de vos propres applications

Vous pouvez générer vos propres applications comme vous générez les modèles d'application.

Générez votre application .NET comme décrit dans la rubrique «Génération des modèles d'application .NET», à la page 19, qui répertorie également les prérequis nécessaires pour générer vos propres applications .NET. Pour plus de conseils sur la manière de générer vos propres applications, utilisez les fichiers makefile fournis avec chaque modèle d'application.

Conseil : Pour vous aider à diagnostiquer les problèmes en cas de panne, vous pouvez compiler vos applications avec les symboles inclus.

Concepts associés

Modèles d'application
Les modèles d'application fournissent une vue d'ensemble des caractéristiques communes de chaque API. Vous pouvez les utiliser pour vérifier votre installation et votre configuration de serveur de messagerie, ou pour vous aider à construire vos propres applications.

Référence associée

.NET interfaces
Cette section documente les interfaces de classe .NET , ainsi que leurs propriétés et leurs méthodes.

Propriétés des objets XMS
Cette chapitre présente les propriétés d'objet définies par XMS.

Reconnexion automatique du client IBM MQ via XMS

Configurez votre client XMS pour vous reconnecter automatiquement après une panne de réseau, de gestionnaire de files d'attente ou de serveur lors de l'utilisation du client IBM WebSphere MQ 7.1 ou version ultérieure en tant que fournisseur de messagerie.

Utilisez les propriétés `WMQ_CONNECTION_NAME_LIST` et `WMQ_CLIENT_RECONNECT_OPTIONS` de la classe`MQConnectionFactory` pour configurer la reconnexion automatique pour une connexion client. Cette fonction permet de reconnecter automatiquement un client après une panne de connexion, ou en tant qu'option après l'arrêt du gestionnaire de files d'attente. La conception de certaines applications client rend celles-ci incompatibles avec la reconnexion automatique.

La reconnexion automatique devient active une fois que la connexion a été établie.

Remarque : Les propriétés `Client Reconnect Options`, `Client Reconnect Timeout` et `Connection Namelist` peuvent également être définies via la table de définition de canal du client ou l'activation de la reconnexion du client dans le fichier `mqclient.ini`.

Remarque : Si les propriétés de reconnexion sont définies sur l'objet `ConnectionFactory` et dans la table de définition de canal du client, la règle de priorité est la suivante. Si la valeur par défaut de la propriété de liste de noms de connexion est définie dans l'objet `ConnectionFactory`, la table de définition de canal du client a priorité. Si la liste de noms de connexion n'est pas définie à sa valeur par défaut, les valeurs de propriété définies dans l'objet `ConnectionFactory` ont la priorité. La valeur par défaut de la liste de noms de connexion est `localhost(1414)`.

Ecriture d'applications XMS .NET

Les rubriques de cette section fournissent des informations qui peuvent vous être utiles si vous écrivez des applications XMS .NET.

Cette section fournit des informations spécifiques à l'écriture d'applications XMS .NET. Pour des informations générales sur l'écriture d'applications XMS, voir «Ecriture d'applications XMS», à la page 20.

Cette section contient les rubriques suivantes :

Concepts associés

Ecriture d'applications XMS
Les rubriques de cette section contiennent des informations d'aide à l'écriture d'applications XMS.

Référence associée

.NET interfaces
Cette section documente les interfaces de classe .NET , ainsi que leurs propriétés et leurs méthodes.

Types de données pour .NET

XMS.NET prend en charge les types System.Boolean, System.Byte, System.SByte, System.Char, System.String, System.Single, System.Double, System.Decimal, System.Int16, System.Int32, System.Int64, System.UInt16, System.UInt32, System.UInt64 et System.Object. Les types de données pour XMS .NET sont différents des types de données pour XMS C/C + +. Vous pouvez utiliser cette rubrique pour identifier les types de données correspondants.

Le tableau suivant présente et décrit brièvement les types de données XMS .NET et XMS C/C++.

Tableau 7. Types de données pour XMS .NET et XMS C/C++

Type XMS .NET	Types XMS C/C++	Description
System.SByte	xmsSBYTE xmsINT8	Valeur 8 bits signée
System.Byte	xmsBYTE xmsUINT8	Valeur 8 bits non signée
System.Int16	xmsINT16 xmsSHORT	Valeur 16 bits signée
System.UInt16	xmsUINT16 xmsUSHORT	Valeur 16 bits non signée
System.Int32	xmsINT32 xmsINT	Valeur 32 bits signée

Tableau 7. Types de données pour XMS .NET et XMS C/C++ (suite)		
Type XMS .NET	**Types XMS C/C++**	**Description**
System.UInt32	xmsUINT32 xmsUINT	Valeur 32 bits non signée
System.Int64	xmsLONG xmsINT64	Valeur 64 bits signée
System.UInt64	xmsULONG xmsUINT64	Valeur 64 bits non signée
System.Char	xmsCHAR16	Caractère 16 bits non signé (Unicode pour .NET)
System.Single	xmsFLOAT	Valeur flottante 32 bits IEEE
System.Double	xmsDOUBLE	Valeur flottante 64 bits IEEE
System.Boolean	xmsBOOL	Valeur True/False
Non applicable	xmsCHAR	Valeur 8 bits signée ou non signée (selon la plateforme)
System.Decimal	Non applicable	Entier 96 bits signé fois 10^0 à 10^{28}
System.Object	Non applicable	Base de tous les types
System.String	Non applicable	Type String

Opérations gérées et non gérées dans .NET

Le code géré est exécuté exclusivement dans la routine d'exécution du langage commun .NET ; il est totalement dépendant des services fournis par cette routine. Une application est classée comme non gérée si une partie de celle-ci exécute ou appelle des services externes à la routine d'exécution du langage commun .NET.

Certaines fonctionnalités avancées ne sont pas prises en charge dans l'environnement géré .NET.

Si votre application requiert une fonctionnalité qui n'est pas prise en charge dans l'environnement géré, vous pouvez modifier l’application de sorte qu'elle utilise l'environnement non géré sans y apporter de modifications substantielles. Toutefois, vous devez prendre en compte le fait que la pile XMS utilise des codes non gérés lorsque cette sélection est effectuée.

Connexions à un gestionnaire de files d'attente IBM MQ

Les connexions gérées à WMQ_CM_CLIENT ne prennent pas en charge les communications non TCP et la compression de canal. Toutefois, ces connexions peuvent être prises en charge via l'utilisation d'une connexion non gérée (WMQ_CM_CLIENT_UNMANAGED). Pour plus d'informations, voir Developing .NET applications.

Si vous créez une fabrique de connexions à partir d'un objet administré dans un environnement non géré, vous devez modifier manuellement la valeur du mode de connexion à XMSC_WMQ_CM_CLIENT_UNMANAGED.

Connexions à un moteur de messagerie de bus d'intégration de services WebSphere Application Server

Les connexions à un moteur de messagerie de bus d'intégration de services WebSphere Application Server nécessitant l'utilisation du protocole SSL (HTTPS y compris) ne sont pas prises en charge en tant que code géré.

Référence associée

XMSC_WMQ_CONNECTION_MODE

Destinations dans .NET

Dans .NET, les destinations sont créées en fonction du type de protocole et peuvent être utilisées uniquement sur le type de protocole pour lequel elles ont été créées.

Deux fonctions permettent de créer des destinations, une pour les rubriques et une pour les files d'attente :

- `IDestination CreateTopic(String topic);`
- `IDestination CreateQueue(String queue);`

Ces fonctions sont disponibles avec les deux objets suivants dans l'API :

- ISession
- XMSFactoryFactory

Dans les deux cas, ces méthodes acceptent une chaîne de style identificateur URI pouvant inclure des paramètres, selon le format suivant :

```
"topic://some/topic/name?priority=5"
```

Ces méthodes acceptent également seulement un nom de destination, c'est-à-dire un nom sans préfixe topic:// ou queue:// ni paramètre.

Ainsi, la chaîne de style identificateur URI

```
CreateTopic("topic://some/topic/name");
```

produit le même résultat que le nom de destination suivant :

```
CreateTopic("some/topic/name");
```

Comme pour le bus d'intégration de services WebSphere Application Server JMS, les rubriques peuvent également être spécifiées dans un format abrégé, qui inclut les éléments *topicname* et *topicspace*, sans paramètre :

```
CreateTopic("topicspace:topicname");
```

Propriétés dans .NET

Une application .NET utilise les méthodes dans l'interface PropertyContext pour extraire et définir les propriétés des objets.

L'interface PropertyContext encapsule les méthodes pour extraire et définir les propriétés. Ces méthodes sont héritées, directement ou indirectement, par les classes suivantes :

- BytesMessage
- Connexion
- ConnectionFactory

- ConnectionMetaData
- Destination
- MapMessage
- Message
- MessageConsumer
- MessageProducer
- ObjectMessage
- QueueBrowser
- Session
- StreamMessage
- TextMessage

Si une application définit la valeur d'une propriété, la nouvelle valeur remplace toute valeur existante.

Pour plus d'informations sur les propriétés XMS, voir «Propriétés des objets XMS», à la page 181.

Dans XMS, par souci de simplification, les noms et les valeurs de propriété XMS sont prédéfinis en tant que constantes publiques dans une structure appelée XMSC. Les noms de ces constantes se présentent sous la forme XMSC.*constante*; par exemple, XMSC.USERID (constante de nom de propriété) et XMSC.DELIVERY_AS_APP (constante de valeur).

De plus, vous pouvez accéder aux constantes IBM MQ par l'intermédiaire de la structure IBM.XMS.MQC. Si l'espace de nom IBM.XMS est déjà importé, vous pouvez accéder aux valeurs de ces propriétés au format MQC.*constante*. Par exemple, MQC.MQRO_COA_WITH_FULL_DATA.

Enfin, si vous disposez d'une application hybride qui utilise à la fois XMS .NET et IBM MQ classes for .NET et qui importe les espaces de nom IBM.XMS et IBM.WMQ, vous devez indiquer le nom qualifié complet de l'espace de nom de structure MQC pour garantir que chaque occurrence est unique.

Certaines fonctionnalités avancées ne sont pas prises en charge dans l'environnement .NET géré. Voir «Opérations gérées et non gérées dans .NET», à la page 46 pour plus de détails.

Gestion des propriétés non existantes dans .NET

La gestion des propriétés non existantes dans XMS .NET est globalement cohérente avec la spécification JMS, mais aussi avec les implémentations C et C++ de XMS.

Dans JMS, l'accès à une propriété non existante peut entraîner une exception système Java lorsqu'une méthode essaie de convertir la valeur non existante (null) dans le type requis. Si une propriété n'existe pas, les exceptions suivantes se produisent :

- getStringProperty et getObjectProperty renvoie une valeur null
- getBooleanProperty renvoie false car Boolean.valueOf(null) renvoie false
- getIntProperty etc. renvoie java.lang.NumberFormatException car Integer.valueOf(null) émet une exception

Si une propriété n'existe pas dans XMS .NET, les exceptions suivantes se produisent :

- GetStringProperty et GetObjectProperty (et GetBytesProperty) renvoient null (identique à Java)
- GetBooleanProperty émet System.NullReferenceException
- GetIntProperty etc. émet System.NullReferenceException

Cette implémentation est différente de Java, mais elle est globalement cohérente avec la spécification JMS et avec les interfaces XMS C et C++. Tout comme l'implémentation Java, XMS .NET propage les exceptions de l'appel System.Convert vers l'appelant. Toutefois, contrairement à Java, XMS émet explicitement NullReferenceExceptions plutôt que d'utiliser le comportement natif de l'infrastructure .NET en transmettant la valeur null aux routines de conversion système. Si votre application définit une chaîne telle que "abc" pour une propriété et appelle GetIntProperty, l'exception System.FormatException émise par Convert.ToInt32("abc") est propagée à l'appelant, ce qui est cohérent

avec Java. L'exception MessageFormatException est émise uniquement si les types utilisés pour setProperty et getProperty ne sont pas compatibles. Ce comportement est également cohérent avec Java.

Traitement des erreurs dans .NET

Les exceptions XMS .NET sont toutes dérivées de System.Exception. Les appels de méthode XMS peuvent émettre des exceptions XMS spécifiques comme MessageFormatException, des exceptions XMSExceptions générales ou des exceptions système comme NullReferenceException.

Ecrivez des applications pour intercepter ces erreurs, soit dans des blocs catch spécifiques soit dans des blocs catch `System.Exception` généraux, en fonction des exigences des applications.

Message et programme d'écoute des exceptions dans .NET

Une application .NET utilise un programme d'écoute de message pour recevoir les messages de manière asynchrone, et un programme d'écoute des exceptions pour être averti de manière asynchrone d'un problème avec une connexion.

La fonctionnalité des programmes d'écoute des messages et des exceptions est la même pour .NET et pour C + +. Toutefois, il existe de petites différences d'implémentation.

Programmes d'écoute de message dans .NET

Pour recevoir des messages de manière asynchrone, vous devez effectuer les étapes suivantes :

1. Définissez une méthode qui correspond à la signature de la délégation du programme d'écoute de message. Vous pouvez définir une méthode statique ou une méthode d'instance dans n'importe quelle classe accessible. La signature de délégation est la suivante :

   ```
   public delegate void MessageListener(IMessage msg);
   ```

   vous pouvez donc définir la méthode de la manière suivante :

   ```
   void SomeMethodName(IMessage msg);
   ```

2. Instanciez cette méthode en tant que délégation, par exemple, de la manière suivante :

   ```
   MessageListener OnMsgMethod = new MessageListener(SomeMethodName)
   ```

3. Enregistrez la délégation avec un ou plusieurs consommateurs en la définissant sur la propriété MessageListener du consommateur :

   ```
   consumer.MessageListener = OnMsgMethod;
   ```

   Vous pouvez annuler la délégation en redéfinissant la valeur Null pour la propriété MessageListener :

   ```
   consumer.MessageListener = null;
   ```

Programmes d'écoute des exceptions dans .NET

Le programme d'écoute des exceptions fonctionne de la même manière que le programme d'écoute de message, mais sa définition de délégation est différente et il est affecté à la connexion plutôt qu'au consommateur de message. Il en est de même pour C++.

1. Définissez la méthode. La signature de délégation est la suivante :

   ```
   public delegate void ExceptionListener(Exception ex);
   ```

vous pouvez donc définir la méthode de la manière suivante :

```
void SomeMethodName(Exception ex);
```

2. Instanciez cette méthode en tant que délégation, par exemple, de la manière suivante :

```
ExceptionListener OnExMethod = new ExceptionListener(SomeMethodName)
```

3. Enregistrez la délégation avec la connexion en définissant sa propriété ExceptionListener :

```
connection.ExceptionListener = OnExMethod ;
```

Vous pouvez annuler la délégation en redéfinissant la valeur Null pour la propriété ExceptionListener :

```
null:  connection.ExceptionListener = null;
```

Lorsqu'il ne reste plus aucune référence aux exceptions ou aux messages, ceux-ci sont supprimés automatiquement par le récupérateur de place du système.

Exemple de code :

```
using System;
using System.Threading;
using IBM.XMS;

public class Sample
{
    public static void Main()
    {
        XMSFactoryFactory factoryFactory = XMSFactoryFactory.GetInstance(XMSC.CT_RTT);

        IConnectionFactory connectionFactory = factoryFactory.CreateConnectionFactory();
        connectionFactory.SetStringProperty(XMSC.RTT_HOST_NAME, "localhost");
        connectionFactory.SetStringProperty(XMSC.RTT_PORT, "1506");

        //
        // Create the connection and register an exception listener
        //

        IConnection connection = connectionFactory.CreateConnection();
        connection.ExceptionListener = new ExceptionListener(Sample.OnException);

        ISession session = connection.CreateSession(false, AcknowledgeMode.AutoAcknowledge);
        IDestination topic = session.CreateTopic("topic://xms/sample");

        //
        // Create the consumer and register an async message listener
        //

        IMessageConsumer consumer = session.CreateConsumer(topic);
        consumer.MessageListener = new MessageListener(Sample.OnMessage);

        connection.Start();

        while (true)
        {
            Console.WriteLine("Waiting for messages....");
            Thread.Sleep(1000);
        }
    }

    static void OnMessage(IMessage msg)
    {
        Console.WriteLine(msg);
    }

    static void OnException(Exception ex)
    {
        Console.WriteLine(ex);
    }
}
```

Utilisation des objets gérés

Les rubriques de cette section fournissent des informations sur les objets gérés. Les applications XMS peuvent extraire des définitions d'objet depuis un référentiel central d'objets gérés et les utiliser pour créer des fabriques de connexions et des destinations.

Cette section fournit des informations qui peuvent vous aider à créer et utiliser des objets gérés ; elle décrit les types de référentiel d'objets gérés pris en charge par XMS. Elle explique également comment une application XMS établit une connexion à un référentiel d'objets gérés afin d'extraire les objets gérés requis.

Cette section contient les rubriques suivantes :

Concepts associés

Objets gérés

Les objets gérés permettent d'administrer les paramètres de connexion utilisés à partir d'un référentiel central. Une application extrait des définitions d'objet depuis le référentiel central et les utilise pour créer des objets `ConnectionFactory` et `Destination`. Les objets gérés permettent de découpler les applications des ressources qu'elles utilisent lors de leur exécution.

Types de référentiel d'objets gérés pris en charge

Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Les répertoires objet de système de fichiers prennent la forme d'objets sérialisés Java Naming Directory Interface (JNDI). Les répertoires objet LDAP sont des répertoires qui contiennent des objets JNDI. Les répertoires objet de système de fichiers et LDAP peuvent être administrés via l'outil JMSAdmin, fourni avec IBM WebSphere MQ 6.0 ou via IBM MQ Explorer, fourni avec IBM WebSphere MQ 7.0 et versions ultérieures. Les répertoires objet de système de fichiers et LDAP permettent d'administrer les connexions client en centralisant les fabriques de connexions et les destinations IBM WebSphere MQ. L'administrateur réseau peut déployer plusieurs applications se référant au même référentiel central, qui sont automatiquement mises à jour pour refléter les modifications de paramètres de connexion effectuées dans le référentiel central.

Un répertoire d'affectation de classe de service contient les fabriques de connexions et les destinations WebSphere Application Server service integration bus ; il peut être administré via la console d'administration WebSphere Application Server. Pour qu'une application XMS extraie des objets depuis un annuaire de dénomination COS, un service Web de recherche JNDI doit être déployé. Ce service Web n'est pas disponible sur toutes les WebSphere Application Server service integration technologies. Pour plus de détails, reportez-vous à la documentation relative au produit.

Remarque : Redémarrez les connexions d'applications pour que les modifications apportées au répertoire d'objet soient prises en compte.

Concepts associés

Mappage de propriété pour les objets gérés

Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Objets gérés
Les objets gérés permettent d'administrer les paramètres de connexion utilisés à partir d'un référentiel central. Une application extrait des définitions d'objet depuis le référentiel central et les utilise pour créer des objets `ConnectionFactory` et `Destination`. Les objets gérés permettent de découpler les applications des ressources qu'elles utilisent lors de leur exécution.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

Mappage de propriété pour les objets gérés

Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Pour créer, par exemple, une fabrique de connexions XMS avec des propriétés extraites d'une fabrique de connexions JMS IBM MQ , les propriétés doivent être mappées entre les deux.

Les mappages de propriétés s'effectuent automatiquement.

Le tableau suivant présente les mappages entre quelques propriétés de fabriques de connexions et de destinations. Ce ne sont que quelques exemples, toutes les propriétés ne sont pas pertinentes pour tous les types de connexion et tous les serveurs.

Tableau 8. Exemples de mappage de noms pour des propriétés de fabrique de connexions et de destination

Nom de la propriété IBM MQ JMS	Nom de la propriété XMS	Nom de la propriété WebSphere Application Server service integration bus
PERSISTENCE (PER)	XMSC_DELIVERY_MODE	
EXPIRY (EXP)	XMSC_TIME_TO_LIVE	
PRIORITY (PRI)	XMSC_PRIORITY	
	XMSC_WPM_HOST_NAME	serverName
	XMSC_WPM_BUS_NAME	busName
	XMSC_WPM_TOPIC_SPACE	topicName

Tableau 9. XMS .NETpropriétés

Propriété	Type d'objet				
	CF	QCF	TCF	File d'attente	Topic
APPLICATIONNAME	Y	Y	Y	N/A	N/A
ASYNCEXCEPTION	Y	Y	Y	N/A	N/A
CCDTURL	Y	Y	Y	N/A	N/A
Canal	Y	Y	Y	N/A	N/A
ConnectionNameList	Y	Y	Y	N/A	N/A
CLIENTRECONNECTOPTIONS	Y	Y	Y	N/A	N/A
CLIENTRECONNECTTIMEOUT	Y	Y	Y	N/A	N/A
clientId	N/A	Y	N/A	N/A	N/A
COMPHDR «1», à la page 54	Y	N/A	Y	N/A	N/A
COMPMSG «1», à la page 54	Y	Y	Y	N/A	N/A
CONNOPT «1», à la page 54	Y	Y	Y	N/A	N/A
CONNTAG «1», à la page 54	Y	Y	Y	N/A	N/A
DESCRIPTION «1», à la page 54	N/A	Y	N/A	Y	Y
Expiration «1», à la page 54	N/A	N/A	N/A	Y	Y
FAILIFQUIESCE	Y	Y	Y	Y	Y
nom_hôte	N/A	Y	N/A	N/A	N/A

Tableau 9. XMS .NETpropriétés (suite)					
Propriété	**Type d'objet**				
	CF	**QCF**	**TCF**	**File d'attente**	**Topic**
LOCALADDRESS	N/A	Y	N/A	N/A	N/A
Persistance	N/A	N/A	N/A	Y	Y
Port	N/A	Y	N/A	N/A	N/A
Priorité[«1», à la page 54]	N/A	N/A	N/A	Y	Y
PROVIDERVERSION [«1», à la page 54]	N/A	Y	N/A	N/A	N/A
QMANAGER	Y	Y	Y	Y	N/A
QUEUE [«1», à la page 54]	N/A	N/A	N/A	Y	N/A
SHARECONVALLOWED	Y	Y	Y	N/A	N/A
TOPIC [«1», à la page 54]	N/A	N/A	N/A	N/A	Y
Transport [«1», à la page 54]	N/A	Y	N/A	N/A	N/A

Remarque :

1. Ces propriétés ne possèdent pas de propriétés au niveau de l'application, mais elles peuvent éventuellement être définies à l'aide de propriétés administrées.

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

Propriétés requises pour les objets ConnectionFactory gérés

Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Les propriétés répertoriées dans les tableaux suivants constituent le minimum requis pour la création d'une connexion à un serveur de messagerie. Si vous voulez personnaliser la manière dont une connexion est créée, votre application peut définir des propriétés supplémentaires pour l'objet ConnectionFactory. Pour plus d'informations, voir la section «Propriétés de ConnectionFactory», à la page 182. Une liste complète des propriétés disponibles est incluse.

Connexion à un gestionnaire de files d'attente IBM MQ

Tableau 10. Paramètres de propriété des objets ConnectionFactory administrés pour les connexions à un gestionnaire de files d'attente IBM MQ

Propriété XMS requise	**Propriété IBM MQ JMS équivalente requise**
XMSC_CONNECTION_TYPE	XMS utilise le nom de classe de la fabrique de connexions et la propriété TRANSPORT (TRAN).
XMSC_WMQ_HOST_NAME	HOSTNAME (HOST)
XMSC_WMQ_PORT	PORT
XMSC_WMQ_QUEUE_MANAGER	Nom du gestionnaire de file d'attente

Connexion en temps réel à un courtier

Tableau 11. Paramètres de propriété pour objets ConnectionFactory gérés pour des connexions en temps réel à un courtier

Propriété XMS requise	Propriété IBM MQ JMS équivalente requise
XMSC_CONNECTION_TYPE	XMS utilise le nom de classe de la fabrique de connexions et la propriété TRANSPORT (TRAN).
XMSC_RTT_HOST_NAME	HOSTNAME (HOST)
XMSC_RTT_PORT	PORT

Connexion à un WebSphere Application Server service integration bus

Tableau 12. Paramètres de propriété pour des objets ConnectionFactory gérés pour des connexions à un WebSphere Application Server service integration bus

Propriété XMS	Description
XMSC_CONNECTION_TYPE	Type de serveur de messagerie auquel une application se connecte.: Cette valeur est déterminée à partir du nom de classe de la fabrique de connexions.
XMSC_WPM_BUS_NAME	Pour une fabrique de connexions, nom du bus d'intégration de services auquel l'application se connecte ; pour une destination, nom du bus d'intégration de services dans lequel la destination existe.

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Connexions sécurisées à un gestionnaire de files d'attente IBM MQ

Pour permettre à une application XMS .NET de créer des connexions sécurisées à un gestionnaire de files d'attente IBM MQ, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Connexions sécurisées à un moteur de messagerie WebSphere Application Server service integration bus
Pour permettre à une application XMS .NET de créer des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

Propriétés requises pour les objets Destination gérés

Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

Tableau 13. Paramètres de propriété pour les objets Destination gérés

Type de connexion	Propriété	Description
Gestionnaire de files d'attente IBM MQ	QUEUE (QU) TOPIC (TOP)	File d'attente à laquelle vous voulez vous connecter Rubrique utilisée par l'application en tant que destination
Connexion en temps réel à un courtier	TOPIC (TOP)	Rubrique utilisée par l'application en tant que destination
WebSphere Application Server service integration bus	topicName queueName	Si votre application se connecte à une rubrique Si votre application se connecte à une file d'attente

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext

Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Création d'objets gérés

La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Avant de commencer

Pour plus de détails sur les différents types de référentiel d'objets gérés pris en charge par XMS, voir «Types de référentiel d'objets gérés pris en charge», à la page 51.

Pourquoi et quand exécuter cette tâche

Afin de créer les objets gérés pour IBM MQ, utilisez IBM MQ Explorer ou l'outil d'administration JMS d'IBM MQ (JMSAdmin).

Pour créer les objets gérés pour IBM MQ ouIBM Integration Bus, utilisez l'outil IBM MQ JMS administration (JMSAdmin).

Pour créer les objets gérés pour WebSphere Application Server service integration bus, utilisez la console d'administrationWebSphere Application Server.

Les étapes suivantes récapitulent la procédure à suivre pour créer des objets gérés.

Procédure

1. Créez une fabrique de connexions et définissez les propriétés nécessaires pour créer une connexion à partir de votre application vers le serveur de votre choix.

   Les propriétés minimales requises par XMS pour créer une connexion sont définies dans «Propriétés requises pour les objets ConnectionFactory gérés», à la page 55.
2. Créez la destination requise sur le serveur de messagerie auquel votre application se connecte :

   - Pour une connexion à un gestionnaire de files d'attente IBM MQ , créez une file d'attente ou une rubrique.
   - Pour une connexion en temps réel à un courtier, créez une rubrique.
   - Pour une connexion à un WebSphere Application Server service integration bus, créez une file d'attente ou une rubrique.

   Les propriétés minimales requises par XMS pour créer une connexion sont définies dans «Propriétés requises pour les objets Destination gérés», à la page 57.

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Objets gérés
Les objets gérés permettent d'administrer les paramètres de connexion utilisés à partir d'un référentiel central. Une application extrait des définitions d'objet depuis le référentiel central et les utilise pour créer des objets `ConnectionFactory` et `Destination`. Les objets gérés permettent de découpler les applications des ressources qu'elles utilisent lors de leur exécution.

Objets ConnectionFactories et Connection
Un objet ConnectionFactory fournit un modèle utilisé par une application pour créer un objet Connection. L'application utilise l'objet Connection pour créer un objet Session.

Connexion à un bus d'intégration de services

Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Tâches associées

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

IDestination (pour l'interface .NET)
Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Propriétés de Destination
Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Objets InitialContext

Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Pourquoi et quand exécuter cette tâche

Un objet InitialContext encapsule une connexion au référentiel. L'API XMS fournit les méthodes pour effectuer les tâches suivantes :

- Créer un objet InitialContext
- Rechercher un objet géré dans le référentiel d'objets gérés.

Pour plus d'informations sur la création d'un objet InitialContext, voir «InitialContext», à la page 113 pour .NET et «Propriétés de InitialContext», à la page 192.

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

InitialContext (pour l'interface .NET)
Une application utilise un objet InitialContext pour créer des objets à partir de définitions d'objet extraits d'un référentiel des objets gérés.

Propriétés de InitialContext
Présentation des propriétés de l'objet InitialContext, avec des liens vers des informations de référence plus détaillées.

Propriétés InitialContext

Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Dans JNDI et dans l'implémentation .NET de XMS, les informations complémentaires sont fournies au constructeur dans une table de hachage d'environnement.

L'emplacement d'un référentiel d'objet géré est défini dans la propriété XMSC_IC_URL. Généralement, cette propriété est transmise sur l'appel Create, mais peut être modifiée de manière à se connecter à un répertoire de désignation avant la recherche. Pour les contextes de système de fichiers et de protocole LDAP, cette propriété définit l'adresse de ce répertoire. Dans le contexte de la dénomination COS, il s'agit de l'adresse du service Web qui utilise ces propriétés pour la connexion au répertoire JNDI.

Les propriétés suivantes sont transmises au service Web, qui les utilise pour établir la connexion au répertoire JNDI :

- XMSC_IC_PROVIDER_URL
- XMSC_IC_SECURITY_CREDENTIALS
- XMSC_IC_SECURITY_AUTHENTICATION
- XMSC_IC_SECURITY_PRINCIPAL
- XMSC_IC_SECURITY_PROTOCOL

Concepts associés

Types de référentiel d'objets gérés pris en charge

Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

InitialContext (pour l'interface .NET)
Une application utilise un objet InitialContext pour créer des objets à partir de définitions d'objet extraits d'un référentiel des objets gérés.

Propriétés de InitialContext
Présentation des propriétés de l'objet InitialContext, avec des liens vers des informations de référence plus détaillées.

Format d'identificateur URI pour contexte initial XMS

L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

Contexte FileSystem

Pour le contexte FileSystem, l'URL indique l'emplacement du répertoire de base du système de fichiers. La structure de l'URL est définie par la norme RFC 1738, *Uniform Resource Locators (URL)* : l'URL est composé du préfixe `file://`, la syntaxe placée après ce préfixe représentant une définition valide d'un fichier pouvant être ouvert sur le système sur lequel XMS est exécuté.

Cette syntaxe peut être spécifique à la plateforme, et utiliser les séparateurs / ou \. Si vous utilisez \, chaque séparateur doit être échappé avec un caractère \ supplémentaire. Ainsi, .NET framework n'essaie pas d'interpréter le séparateur en tant que caractère d'échappement pour ce qui suit.

Ces exemples illustrent cette syntaxe :

```
file://myBindings
file:///admin/.bindings
file://\\admin\\.bindings
file://c:/admin/.bindings
file://c:\\admin\\.bindings
file://\\\\madison\\shared\\admin\\.bindings
file:///usr/admin/.bindings
```

Contexte LDAP

Pour le contexte LDAP, la structure de base de l'URL est conforme à la norme RFC 2255, *The LDAP URL Format*, avec le préfixe non sensible à la casse `ldap://`

La syntaxe exacte est illustrée dans l'exemple suivant :

```
LDAP://[Hostname][:Port]["/"[DistinguishedName]]
```

Cette syntaxe est conforme à la norme RFC, mais sans prise en charge d'attributs, de portée, de filtres ou d'extensions.

Exemples de syntaxe :

```
ldap://madison:389/cn=JMSData,dc=IBM,dc=UK
ldap://madison/cn=JMSData,dc=IBM,dc=UK
LDAP:///cn=JMSData,dc=IBM,dc=UK
```

Contexte WSS

Pour le contexte WSS, l'URL se présente sous la forme d'un noeud final de services Web, avec le préfixe `http://`.

Vous pouvez également utiliser le préfixe `cosnaming://` ou `wsvc://`.

Ces deux préfixes signifient que vous utilisez un contexte WSS avec l'URL en accès via http, ce qui permet au type de contexte initial d'être facilement déduit directement à partir de l'URL.

Exemples de syntaxe :

```
http://madison.ibm.com:9080/xmsjndi/services/JndiLookup
cosnaming://madison/jndilookup
```

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au

référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

InitialContext (pour l'interface .NET)
Une application utilise un objet InitialContext pour créer des objets à partir de définitions d'objet extraits d'un référentiel des objets gérés.

Propriétés de InitialContext
Présentation des propriétés de l'objet InitialContext, avec des liens vers des informations de référence plus détaillées.

service Web de recherche JNDI

Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Le service Web est fourni dans le fichier d'archive d'entreprise SIBXJndiLookupEAR.ear, qui se trouve dans le répertoire d'installation. Pour la version actuelle de IBM Message Service Client for .NET, SIBXJndiLookupEAR.ear se trouve dans le répertoire *install_dir*\java\lib . Il peut être installé sur un serveur de WebSphere Application Server service integration bus via la console d'administration ou l'outil de création de script wsaadmin. Reportez-vous à la documentation du produit pour plus d'informations sur le déploiement d'applications de service Web.

Pour définir le service Web dans des applications XMS, il suffit de définir la propriété XMSC_IC_URL de l'objet InitialContext sur l'URL du point d'extrémité de service Web. Par exemple, si le service Web est déployé sur le serveur hôte MyServer, vous pouvez définir l'adresse URL de noeud final de service Web suivante :

```
wsvc://MyHost:9080/SIBXJndiLookup/services/JndiLookup
```

La définition de la propriété XMSC_IC_URL permet à InitialContext Lookup d'appeler le service Web sur le noeud final défini, qui à son tour recherche l'objet géré requis à partir du service de dénomination COS.

Les applications .NET peuvent utiliser le service Web. Le déploiement côté serveur est identique pour XMS C ou /C++, et pour XMS .NET.XMS .NET appelle le service Web directement via l'infrastructure Microsoft .NET .

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Configuration de l'environnement d'un serveur de messagerie
Les rubriques de cette section expliquent comment configurer l'environnement d'un serveur de messagerie pour permettre aux applications XMS de se connecter à un serveur.

Utilisation des modèles d'application XMS
Utilisez les modèles d'application fournis avec XMS pour vérifier l'installation et la configuration du serveur de messagerie ; ces modèles vous aident également à générer vos propres applications. Les modèles fournissent une présentation des caractéristiques communes de chaque API.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Installation de Message Service Client for .NET à l'aide de l'assistant d'installation
L'installation utilise le programme d'installation InstallShield X/Windows MSI. Vous avez le choix entre une installation complète ou personnalisée.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

Extraction d'objets gérés

XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Les objets pouvant être extraits peuvent avoir les types de nom suivants :

- Un nom simple décrivant l'objet Destination, par exemple, une destination de file d'attente appelée SalesOrders
- Un nom composé qui peut être constitué d'éléments SubContexts, séparés par le caractère /, et doit se terminer par le nom d'objet. Exemple de nom composé : "Warehouse/PickLists/DispatchQueue2", où Warehouse et Picklists sont des éléments SubContexts du répertoire de désignation et DispatchQueue2, le nom d'un objet Destination.

Concepts associés

Types de référentiel d'objets gérés pris en charge
Les objets gérés de type système de fichiers et protocole LDAP peuvent être utilisés pour se connecter à IBM MQ et à WebSphere Application Server, tandis que le répertoire d'affectation de classe de service permet de se connecter uniquement à WebSphere Application Server.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

service Web de recherche JNDI
Pour que vous puissiez accéder à un annuaire de dénomination COS depuis XMS, un service Web de recherche JNDI doit être déployé sur un serveur de WebSphere Application Server service integration bus. Ce service Web transforme les informations Java du service de dénomination COS dans un format lisible par les applications t XMS.

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

InitialContext (pour l'interface .NET)
Une application utilise un objet InitialContext pour créer des objets à partir de définitions d'objet extraits d'un référentiel des objets gérés.

Propriétés de InitialContext
Présentation des propriétés de l'objet InitialContext, avec des liens vers des informations de référence plus détaillées.

Sécurisation des communications pour les applications XMS

Cette section fournit des informations sur la définition de communications sécurisées afin de permettre aux applications XMS de se connecter via SSL (Secure Sockets Layer) à un moteur de messagerie de WebSphere Application Server service integration bus ou à un gestionnaire de files d'attente IBM MQ.

Cette section contient les rubriques suivantes :

- «Connexions sécurisées à un gestionnaire de files d'attente IBM MQ», à la page 67
- «Mappages des noms de suite de chiffrement et de CipherSpec pour les connexions à un gestionnaire de files d'attente IBM MQ», à la page 68
- «Connexions sécurisées à un moteur de messagerie WebSphere Application Server service integration bus», à la page 69
- «Mappages de nom CipherSuite et CipherSpec pour connexions à un WebSphere Application Server service integration bus», à la page 70

Connexions sécurisées à un gestionnaire de files d'attente IBM MQ

Pour permettre à une application XMS .NET de créer des connexions sécurisées à un gestionnaire de files d'attente IBM MQ, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Le protocole utilisé dans la négociation de chiffrement peut être SSL (Secure Sockets Layer) ou TLS (Transport Layer Security), selon la suite de chiffrement CipherSuite spécifiée dans l'objet ConnectionFactory.

Si vous utilisez les bibliothèques client IBM WebSphere MQ 7.0.0 Fix Pack 1 et ultérieures et que vous vous connectez à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , vous pouvez créer plusieurs connexions à un même gestionnaire de files d'attente dans l'application XMS . Toutefois, la connexion à un autre gestionnaire de files d'attente n'est pas autorisée. Dans ce cas, vous recevez une erreur `MQRC_SSL_ALREADY_INITIALIZED`.

Si vous utilisez des bibliothèques client IBM WebSphere MQ 6.0 ou ultérieure, vous pouvez créer une connexion SSL seulement après avoir fermé toute connexion SSL existante. Les connexions SSL simultanées depuis un même processus vers un ou plusieurs gestionnaires de files d'attente ne sont pas autorisées. Si vous effectuez plusieurs demandes, vous recevez l'avertissement `MQRC_SSL_ALREADY_INITIALIZED`, ce qui signifie que certains paramètres ont été ignorés pour la connexion SSL.

Les propriétés ConnectionFactory pour les connexions via SSL à un gestionnaire de files d'attente IBM MQ, ainsi qu'une brève description, sont présentées dans le tableau suivant :

Tableau 14. Propriétés de ConnectionFactory pour des connexions à un gestionnaire de files d'attente IBM MQ via SSL

Nom de la propriété	Description
XMSC_WMQ_SSL_CERT_STORES	Emplacements des serveurs contenant les listes de révocation de certificat à utiliser sur une connexion SSL à un gestionnaire de files d'attente.
XMSC_WMQ_SSL_CIPHER_SPEC	Nom de la spécification CipherSpec à utiliser sur une connexion sécurisée vers un gestionnaire de files d'attente.
XMSC_WMQ_SSL_CIPHER_SUITE	Nom de la suite de chiffrement à utiliser sur une connexion TLS à un gestionnaire de files d'attente. Le protocole utilisé lors de la négociation de la connexion sécurisée dépend de la suite de chiffrement spécifiée.

Tableau 14. Propriétés de ConnectionFactory pour des connexions à un gestionnaire de files d'attente IBM MQ via SSL (suite)	
Nom de la propriété	**Description**
XMSC_WMQ_SSL_CRYPTO_HW	Détails de configuration pour le matériel de cryptographie connecté au système client.
XMSC_WMQ_SSL_FIPS_REQUIRED	La valeur de cette propriété détermine si une application peut ou non utiliser des suites de chiffrement conformes non FIPS. Si cette propriété est pour valeur true, seuls les algorithmes FIPS sont utilisés pour la connexion client/serveur.
XMSC_WMQ_SSL_KEY_REPOSITORY	Emplacement du fichier de clés dans lequel les clés et les certificats sont stockés.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Nombre total d'octets non chiffrés envoyés et reçus dans une conversation SSL avant renégociation de la clé confidentielle.
XMSC_WMQ_SSL_PEER_NAME	Nom d'homologue à utiliser sur une connexion SSL vers un gestionnaire de files d'attente.

Référence associée

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Mappages des noms de suite de chiffrement et de CipherSpec pour les connexions à un gestionnaire de files d'attente IBM MQ

InitialContext effectue la conversion entre la propriété de fabrique de connexions JMSAdmin SSLCIPHERSUITE et la propriété XMS XMSC_WMQ_SSL_CIPHER_SPEC la plus proche. Une conversion similaire est nécessaire si vous spécifiez une valeur pour XMSC_WMQ_SSL_CIPHER_SUITE et aucune valeur pour XMSC_WMQ_SSL_CIPHER_SPEC.

Le Tableau 15, à la page 68 liste les spécifications CipherSpec disponibles et les algorithmes JSSE CipherSuite équivalents.

Tableau 15. Spécifications CipherSpec disponibles et algorithmes JSSE CipherSuite équivalents	
CipherSpec	**Algorithme JSSE CipherSuite équivalent**
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA

Remarque :

- TLS_RSA_WITH_3DES_EDE_CBC_SHA a été déprécié. Toutefois, vous pouvez tout de même l'utiliser pour transférer jusqu'à 32 Go de données avant que la connexion ne soit arrêtée avec l'erreur AMQ9288. Pour éviter cette erreur, n'utilisez pas la norme DES triple ou activez la réinitialisation de clé confidentielle lors de l'utilisation de ce CipherSpec.

Connexions sécurisées à un moteur de messagerie WebSphere Application Server service integration bus

Pour permettre à une application XMS .NET de créer des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

XMS fournit la prise en charge de SSL et HTTPS pour les connexions à un bus d'intégration de services WebSphere. SSL et HTTPS assurent des connexions sécurisées pour l'authentification et la confidentialité.

Tout comme la sécurité WebSphere, la sécurité XMS est configurée en conformité avec les normes et les conventions de dénomination de sécurité JSSE, qui incluent l'utilisation de suites de chiffrement CipherSuites pour la spécification des algorithmes utilisés lors de la négociation d'une connexion sécurisée. Le protocole utilisé lors de la négociation de chiffrement peut être SSL ou TLS, selon la suite de chiffrement CipherSuite spécifiée dans l'objet ConnectionFactory.

Le Tableau 16, à la page 69 répertorie les propriétés qui doivent être définies dans l'objet ConnectionFactory.

Tableau 16. Propriétés de ConnectionFactory pour des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus

Nom de la propriété	Description
XMSC_WPM_SSL_CIPHER_SUITE	Nom de la suite de chiffrement à utiliser pour une connexion TLS à un moteur de messagerie de bus d'intégration de services WebSphere. Le protocole utilisé lors de la négociation de la connexion sécurisée dépend de la suite de chiffrement spécifiée.
XMSC_WPM_SSL_KEYRING_LABEL	Certificat à utiliser lors de l'authentification avec le serveur.

Voici un exemple de propriétés ConnectionFactory pour des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus :

```
cf.setStringProperty(XMSC_WPM_PROVIDER_ENDPOINTS, host_name:port_number:chain_name);
cf.setStringProperty(XMSC_WPM_SSL_KEY_REPOSITORY, key_repository_pathname);
cf.setStringProperty(XMSC_WPM_TARGET_TRANSPORT_CHAIN, transport_chain);
cf.setStringProperty(XMSC_WPM_SSL_CIPHER_SUITE, cipher_suite);
cf.setStringProperty(XMSC_WPM_SSL_KEYRING_STASH_FILE, stash_file_pathname);
```

Où nom_chaîne doit être BootstrapTunneledSecureMessaging ou BootstrapSecureMessaging, et numéro_port représente le numéro de port sur lequel le serveur d'amorçage est à l'écoute des demandes entrantes.

Voici un exemple de propriétés ConnectionFactory avec des valeurs pour des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus :

```
  /* CF properties needed for an SSL connection */
    cf.setStringProperty(XMSC_WPM_PROVIDER_ENDPOINTS,"localhost:7286:BootstrapSecureMessaging");
    cf.setStringProperty(XMSC_WPM_TARGET_TRANSPORT_CHAIN,"InboundSecureMessaging");
    cf.setStringProperty(XMSC_WPM_SSL_KEY_REPOSITORY,"C:\\Program Files\\IBM\\gsk7\\bin\
\XMSkey.kdb");
    cf.setStringProperty(XMSC_WPM_SSL_KEYRING_STASH_FILE,"C:\\Program Files\\IBM\\gsk7\\bin\
\XMSkey.sth");
    cf.setStringProperty(XMSC_WPM_SSL_CIPHER_SUITE,"SSL_RSA_EXPORT_WITH_RC4_40_MD5");
```

Référence associée

IConnectionFactory (pour l'interface .NET)
Une application utilise une fabrique de connexions pour créer une connexion.

Propriétés de ConnectionFactory
Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Mappages de nom CipherSuite et CipherSpec pour connexions à un WebSphere Application Server service integration bus

GSKit utilisant les spécifications CipherSpec plutôt que les algorithmes CipherSuite, les noms CipherSuite de style JSSE spécifiés dans la propriété XMSC_WPM_SSL_CIPHER_SUITE doivent être mappés vers des noms CipherSpec de style GSKit.

Le Tableau 17, à la page 70 liste la spécification CipherSpec équivalente pour chaque algorithme CipherSuite reconnu.

Tableau 17. Algorithmes CipherSuite disponibles et spécifications CipherSpec équivalentes

Algorithme de cryptographie	**Spécification CipherSpec équivalente**
TLS_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA

Remarque :

- TLS_RSA_WITH_3DES_EDE_CBC_SHA a été déprécié. Toutefois, vous pouvez tout de même l'utiliser pour transférer jusqu'à 32 Go de données avant que la connexion ne soit arrêtée avec l'erreur AMQ9288. Pour éviter cette erreur, n'utilisez pas la norme DES triple ou activez la réinitialisation de clé confidentielle lors de l'utilisation de ce CipherSpec.

Messages XMS

Cette section décrit la structure et le contenu des messages XMS et explique comment les applications traitent les messages XMS.

Cette section contient les rubriques suivantes :

- «Parties d'un message XMS», à la page 70
- «Zones d'en-tête dans un message XMS», à la page 71
- «Propriétés d'un message XMS», à la page 72
- «Corps d'un message XMS», à la page 76
- «Sélecteurs de message», à la page 82
- «Mappage de messages XMS en messages IBM MQ», à la page 83

Référence associée

IMessage (pour l'interface .NET)
Un objet Message représente un message envoyé ou reçu par une application. IMessage est une superclasse des classes de messages comme IMapMessage.

Parties d'un message XMS

Un message XMS est composé d'un en-tête, d'un ensemble de propriétés et d'un corps.

En-tête

L'en-tête d'un message contient des zones ; tous les messages contiennent le même ensemble de zones d'en-tête. XMS et les applications utilisent les valeurs des zones d'en-tête pour identifier et acheminer les messages. Pour plus d'informations sur les zones d'en-tête, voir «Zones d'en-tête dans un message XMS», à la page 71.

Ensemble de propriétés
Les propriétés d'un message spécifient des informations supplémentaires sur celui-ci. Bien que tous les messages disposent du même ensemble de zones d'en-tête, chaque message peut avoir un ensemble de propriétés différent. Pour plus d'informations, voir la section «Propriétés d'un message XMS», à la page 72.

Corps
Le corps d'un message contient des données d'application. Pour plus d'informations, voir la section «Corps d'un message XMS», à la page 76.

Une application peut sélectionner les messages qu'elle souhaite recevoir. En utilisant des sélecteurs de message, qui spécifient les critères de sélection. Les critères peuvent être basés sur les valeurs de certaines zones d'en-tête et sur les valeurs de l'une des propriétés du message. Pour plus d'informations sur les sélecteurs de message, voir «Sélecteurs de message», à la page 82.

Référence associée

Zones d'en-tête dans un message XMS
Pour permettre à une application XMS d'échanger des messages avec une application WebSphere JMS, l'en-tête d'un message XMS contient les zones d'en-tête du message JMS.

Propriétés d'un message XMS
XMS prend en charge trois sortes de propriétés de message : les propriétés définies par JMS, les propriétés définies par IBM et les propriétés définies par l'application.

Corps d'un message XMS
Le corps d'un message contient des données d'application. Un message peut toutefois ne pas comporter de corps, mais uniquement des zones d'en-tête et des propriétés.

Sélecteurs de message
Une application XMS utilise des sélecteurs de message pour sélectionner les messages qu'elle souhaite recevoir.

Mappage de messages XMS en messages IBM MQ
Les zones d'en-tête et les propriétés JMS d'un message XMS sont mappées vers des zones dans les structures d'en-tête d'un message IBM MQ.

Zones d'en-tête dans un message XMS

Pour permettre à une application XMS d'échanger des messages avec une application WebSphere JMS, l'en-tête d'un message XMS contient les zones d'en-tête du message JMS.

Les noms de ces zones d'en-tête commencent par le préfixe JMS. Pour une description des zones d'en-tête de message JMS, voir la*spécification Java Message Service*.

XMS implémente les zones d'en-tête de message JMS en tant qu'attributs d'un objet Message. Chaque zone d'en-tête a ses propres méthodes de définition et d'extraction de sa valeur. Pour une description de ces méthodes, voir «IMessage», à la page 126. Une zone d'en-tête est toujours accessible en lecture et en écriture.

Le Tableau 18, à la page 71 répertorie les zones d'en-tête de message JMS et indique comment la valeur de chaque zone est définie pour un message transmis. Certaines zones sont définies automatiquement par XMS lorsqu'une application envoie un message ou, dans le cas de JMSRedelivered, lorsqu'une application reçoit un message.

Tableau 18. Zones d'en-tête de message JMS

Nom de la zone d'en-tête de message JMS	**Mode de définition de la valeur pour un message transmis (au format *méthode* [*classe*])**
JMSCorrelationID	Set JMSCorrelationID [Message]
JMSDeliveryMode	Send [MessageProducer]
JMSDestination	Send [MessageProducer]

Tableau 18. Zones d'en-tête de message JMS (suite)	
Nom de la zone d'en-tête de message JMS	**Mode de définition de la valeur pour un message transmis (au format *méthode* [*classe*])**
JMSExpiration	Send [MessageProducer]
JMSMessageID	Send [MessageProducer]
JMSPriority	Send [MessageProducer]
JMSRedelivered	Receive [MessageConsumer]
JMSReplyTo	Set JMSReplyTo [Message]
JMSTimestamp	Send [MessageProducer]
JMSType	Set JMSType [Message]

Référence associée

Parties d'un message XMS
Un message XMS est composé d'un en-tête, d'un ensemble de propriétés et d'un corps.

Propriétés d'un message XMS
XMS prend en charge trois sortes de propriétés de message : les propriétés définies par JMS, les propriétés définies par IBM et les propriétés définies par l'application.

Corps d'un message XMS
Le corps d'un message contient des données d'application. Un message peut toutefois ne pas comporter de corps, mais uniquement des zones d'en-tête et des propriétés.

Sélecteurs de message
Une application XMS utilise des sélecteurs de message pour sélectionner les messages qu'elle souhaite recevoir.

Mappage de messages XMS en messages IBM MQ
Les zones d'en-tête et les propriétés JMS d'un message XMS sont mappées vers des zones dans les structures d'en-tête d'un message IBM MQ.

Propriétés d'un message XMS

XMS prend en charge trois sortes de propriétés de message : les propriétés définies par JMS, les propriétés définies par IBM et les propriétés définies par l'application.

Une application XMS peut échanger des messages avec une applicationWebSphere JMS car XMS prend en charge les propriétés prédéfinies d'un objet Message suivantes :

- Les mêmes propriétés définies par JMS que celles prises en charge par WebSphere JMS. Les noms de ces propriétés commencent par le préfixe JMSX.
- Les mêmes propriétés définies par IBM que celles prises en charge par WebSphere JMS. Les noms de ces propriétés commencent par le préfixe JMS_IBM_.

Chaque propriété prédéfinie a deux noms :

- Un nom JMS pour une propriété définie par JMS, ou un nom WebSphere JMS pour une propriété définie par IBM.

  Il s'agit du nom sous lequel la propriété est connue dans JMS ou WebSphere JMS, et également du nom transmis avec un message comportant cette propriété. Une application XMS utilise ce nom pour identifier la propriété dans une expression de sélecteur de message.
- Un nom XMS pour identifier la propriété dans toutes les situations, sauf dans une expression de sélecteur de message. Chaque nom XMS est défini en tant que constante nommée dans la classe `IBM.XMS.XMSC`. La valeur de la constante nommée est le nom JMS ou WebSphere JMS correspondant.

En complément des propriétés prédéfinies, une application XMS peut créer et utiliser son propre ensemble de propriétés de message. Ces propriétés sont appelées *propriétés définies par l'application.*

Les propriétés d'un message créé par une application sont accessibles en lecture et en écriture. Elles le reste après l'envoi du message par l'application. Lorsqu'une application reçoit un message, les propriétés de ce message sont en lecture seule. Si une application appelle la méthode `Clear Properties` de la classe Message alors que les propriétés d'un message sont en lecture seule, celles-ci deviennent accessibles en lecture et en écriture. La méthode efface également les propriétés.

Le message reçu, puis réacheminé après suppression de ses propriétés, se comporte de manière cohérente avec le comportement de réacheminement d'un BytesMessage MQ XMS for .NET standard dont les propriétés sont effacées.

Ceci n'est toutefois pas recommandé, car les propriétés suivante seront perdues :

- La valeur de la propriété JMS_IBM_Encoding, ce qui implique que les données de message ne peuvent pas être décodées correctement.
- La valeur de la propriété JMS_IBM_Format, ce qui implique que le chaînage d'en-tête entre l'en-tête de message (MQMD ou le nouveau MQRFH2) et les en-têtes existants sera interrompu.

Pour déterminer les valeurs de toutes les propriétés d'un message, une application peut appeler la méthode Get Properties de la classe Message. La méthode crée un itérateur qui encapsule une liste d'objets Property, où chaque objet représente une propriété du message. L'application peut ensuite utiliser les méthodes de la classe Iterator pour extraire tout à tour chaque objet Property, mais aussi les méthodes de la classe Property pour extraire le nom, le type de données et la valeur de chaque propriété.

Référence associée

Parties d'un message XMS
Un message XMS est composé d'un en-tête, d'un ensemble de propriétés et d'un corps.

Zones d'en-tête dans un message XMS
Pour permettre à une application XMS d'échanger des messages avec une application WebSphere JMS, l'en-tête d'un message XMS contient les zones d'en-tête du message JMS.

Corps d'un message XMS
Le corps d'un message contient des données d'application. Un message peut toutefois ne pas comporter de corps, mais uniquement des zones d'en-tête et des propriétés.

Sélecteurs de message
Une application XMS utilise des sélecteurs de message pour sélectionner les messages qu'elle souhaite recevoir.

Mappage de messages XMS en messages IBM MQ
Les zones d'en-tête et les propriétés JMS d'un message XMS sont mappées vers des zones dans les structures d'en-tête d'un message IBM MQ.

Propriétés de message définies par JMS

Plusieurs propriétés de message définies par JMS sont prises en charge à la fois par XMS et parWebSphere JMS.

Tableau 19, à la page 74 liste les propriétés de message définies par JMS prises en charge à la fois par XMS et par WebSphere JMS. Pour une description des propriétés définies par JMS, voir la *spécification Java Message Service*. Les propriétés définies par JMS ne sont pas valides pour une connexion en temps réel à un courtier.

Le tableau spécifie le type de données de chaque propriété et indique la manière dont la valeur de la propriété est définie pour un message transmis. Certaines propriétés sont définies automatiquement par XMS lors de l'envoi d'un message par une application ou, dans le cas de JMSXDeliveryCount, lors de la réception d'un message par une application.

Tableau 19. Propriétés de message définies par JMS			
Nom XMS de la propriété définie par JMS	**Nom JMS**	**Type de données**	**Mode de définition de la valeur pour un message transmis (au format *méthode* [*classe*])**
JMSX_APPID	JMSXAppID	System.String	Send [MessageProducer]
JMSX_DELIVERY_COUNT	JMSXDeliveryCount	System.Int32	Receive [MessageConsumer]
JMSX_GROUPID	JMSXGroupID	System.String	Set String Property [PropertyContext]
JMSX_GROUPSEQ	JMSXGroupSeq	System.Int32	Set Integer Property [PropertyContext]
JMSX_USERID	JMSXUserID	System.String	Send [MessageProducer]

Propriétés de message définies par IBM

Plusieurs propriétés de message définies par IBM sont prises en charge par XMS et WebSphere JMS.

Tableau 20, à la page 74 répertorie les propriétés de message définies par IBM qui sont prises en charge par XMS et par WebSphere JMS. Pour plus d'informations sur les propriétés définies par IBM, voir la documentation du produit IBM MQ ou WebSphere Application Server.

Le tableau spécifie le type de données de chaque propriété et indique la manière dont la valeur de la propriété est définie pour un message transmis. Certaines propriétés sont définies automatiquement par XMS lors de l'envoi d'un message par une application.

Tableau 20. Propriétés de message définies par IBM			
Nom XMS de la propriété définie par IBM	**Nom WebSphere JMS**	**Type de données**	**Mode de définition de la valeur pour un message transmis (au format *méthode* [*classe*])**
JMS_IBM_CHARACTER_SET	JMS_IBM_Character_Set	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_ENCODING	JMS_IBM_Encoding	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_EXCEPTIONMESSAGE	JMS_IBM_ExceptionMessage	System.String	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONREASON	JMS_IBM_ExceptionReason	System.Int32	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONTIMESTAMP	JMS_IBM_ExceptionTimestamp	System.Int64	Receive [MessageConsumer]
JMS_IBM_EXCEPTIONPROBLEM DESTINATION	JMS_IBM_ExceptionProblemDestination	System.String	Receive [MessageConsumer]
JMS_IBM_FEEDBACK	JMS_IBM_Feedback	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_FORMAT	JMS_IBM_Format	System.String	Set String Property [PropertyContext]

Tableau 20. Propriétés de message définies par IBM (suite)

Nom XMS de la propriété définie par IBM	Nom WebSphere JMS	Type de données	Mode de définition de la valeur pour un message transmis (au format *méthode* [*classe*])
JMS_IBM_LAST_MSG_IN_GROUP	JMS_IBM_Last_Msg_In_Group	System.Boolean	Set Integer Property [PropertyContext]
JMS_IBM_MSGTYPE	JMS_IBM_MsgType	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_PUTAPPLTYPE	JMS_IBM_PutApplType	System.Int32	Send [MessageProducer]
JMS_IBM_PUTDATE	JMS_IBM_PutDate	System.String	Send [MessageProducer]
JMS_IBM_PUTTIME	JMS_IBM_PutTime	System.String	Send [MessageProducer]
JMS_IBM_REPORT_COA	JMS_IBM_Report_COA	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_COD	JMS_IBM_Report_COD	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_DISCARD_MSG	JMS_IBM_Report_Discard_Msg	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_EXCEPTION	JMS_IBM_Report_Exception	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_EXPIRATION	JMS_IBM_Report_Expiration	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_NAN	JMS_IBM_Report_NAN	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_PAN	JMS_IBM_Report_PAN	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_PASS_CORREL_ID	JMS_IBM_Report_Pass_Correl_ID	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_REPORT_PASS_MSG_ID	JMS_IBM_Report_Pass_Msg_ID	System.Int32	Set Integer Property [PropertyContext]
JMS_IBM_SYSTEM_MESSAGEID	JMS_IBM_System_MessageID	System.String	Send [MessageProducer]

Propriétés définies par l'application d'un message

Une application XMS peut créer et utiliser son propre ensemble de propriétés de message. Lorsqu'une application envoie un message, ces propriétés sont également transmises. Une application de réception peut, à l'aide de sélecteurs de message, choisir les messages qu'elle veut recevoir en fonction de la valeur de ces propriétés.

Pour permettre à une application WebSphere JMS de sélectionner et de traiter les messages envoyés par une application XMS , le nom d'une propriété définie par l'application doit être conforme aux règles de formation des identificateurs dans les expressions de sélecteur de message, comme indiqué dans la documentation du produit IBM MQ . Le type de données d'une valeur de propriété définie par l'application doit être l'un des suivants : System.Boolean, System.SByte, System.Int16, System.Int32, System.Int64, System.Float, System.Double ou System.String.

Corps d'un message XMS

Le corps d'un message contient des données d'application. Un message peut toutefois ne pas comporter de corps, mais uniquement des zones d'en-tête et des propriétés.

XMS prend en charge cinq types de corps de message :

Octets
Le corps contient un flux d'octets. Un message avec ce type de corps est appelé un *message d'octets.* L'interface IBytesMessage contient les méthodes permettant de traiter le corps d'un message d'octets. Pour plus d'informations, voir «Messages d'octets», à la page 78.

Mapper
Le corps contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données. Un message avec ce type de corps est appelé un *message de mappe.* L'interface IMapMessage contient les méthodes permettant de traiter le corps d'un message de mappe. Pour plus d'informations, voir «Messages de mappe», à la page 78.

Objet
Le corps contient un objet Java ou .NET sérialisé. Un message avec ce type de corps est appelé un *message d'objet.* L'interface IObjectMessage contient les méthodes permettant de traiter le corps d'un message d'objet. Pour plus d'informations, voir «Messages d'objet», à la page 79.

Flux
Le corps contient un flux de valeurs, où chaque valeur est associée à un type de données. Un message avec ce type de corps est appelé un *message de flux.* L'interface IStreamMessage contient les méthodes permettant de traiter le corps d'un message de flux. Pour plus d'informations, voir «Messages de flux», à la page 80.

Texte
Le corps contient une chaîne. Un message avec ce type de corps est appelé un *message texte.* L'interface ITextMessage contient les méthodes permettant de traiter le corps d'un message texte. Pour plus d'informations, voir «Messages texte», à la page 81.

L'interface IMessage est le parent de tous les objets message et peut être utilisée dans les fonctions de messagerie pour représenter tous les types de message XMS.

Pour plus d'informations sur la taille et les valeurs maximales et minimales de chacun de ces types de données, voir Tableau 5, à la page 40.

Référence associée

Parties d'un message XMS
Un message XMS est composé d'un en-tête, d'un ensemble de propriétés et d'un corps.

Zones d'en-tête dans un message XMS
Pour permettre à une application XMS d'échanger des messages avec une application WebSphere JMS, l'en-tête d'un message XMS contient les zones d'en-tête du message JMS.

Propriétés d'un message XMS
XMS prend en charge trois sortes de propriétés de message : les propriétés définies par JMS, les propriétés définies par IBM et les propriétés définies par l'application.

Sélecteurs de message
Une application XMS utilise des sélecteurs de message pour sélectionner les messages qu'elle souhaite recevoir.

Mappage de messages XMS en messages IBM MQ
Les zones d'en-tête et les propriétés JMS d'un message XMS sont mappées vers des zones dans les structures d'en-tête d'un message IBM MQ.

Types de données d'éléments de données d'application

Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Pour cette raison, le type de données de chaque élément de données d'application écrit dans le corps d'un message par une application XMS doit être l'un des types de données répertoriés dans le Tableau 21, à la page 77. Pour chaque type de données, le tableau présente le type de données Java compatible. XMS fournit les méthodes qui permettent d'écrire des éléments de données d'application seulement avec ces types de données.

Tableau 21. Types de données XMS compatibles avec les types de données Java

Type de données XMS	Représentation	Type de données Java compatible
System.Boolean	Valeur booléenne true ou false	boolean
System.Char16	Caractère codé sur deux octets	Caractère
System.SByte	Entier 8 bits signé	byte
System.Int16	Entier 16 bits signé	court
System.Int32	Entier 32 bits signé	int
System.Int64	Entier 64 bits signé	long
System.Float	Nombre en virgule flottante signé	float
System.Double	Nombre en virgule flottante à double précision signé	double
System.String	Chaîne de caractères	String

Pour plus d'informations sur la taille et les valeurs maximales et minimales de chacun de ces types de données, voir «Types primitifs XMS», à la page 40.

Concepts associés

Attributs et propriétés des objets
Un objet XMS peut être caractérisé par des attributs et des propriétés, implémentés de différentes manières.

Types primitifs XMS
XMS fournit des équivalents pour les huit types primitifs Java (byte, short, int, long, float, double, char et boolean). Ils permettent l'échange de messages entre XMS et JMS sans perte ni corruption de données.

Conversion implicite d'une valeur de propriété d'un type de données à un autre
Lorsqu'une application extrait la valeur d'une propriété, celle-ci peut être convertie par XMS dans un autre type de données. De nombreuses règles régissent les conversions prises en charge et la manière dont XMS les réalise.

Référence associée

Messages d'octets
Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Messages d'objet
Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

Messages de flux
Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Messages texte
Le corps d'un message texte contient une chaîne.

Messages d'octets

Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

Les messages d'octets sont utiles si l'application XMS a besoin d'échanger des messages avec des applications qui n'utilisent pas l'interface de programme d'application XMS ou JMS.

Lorsqu'une application a créé un message d'octets, le corps de ce message est accessible en écriture seule. L'application assemble les données d'application dans le corps en appelant les méthodes d'écriture appropriées de l'interface IBytesMessage pour .NET. Chaque fois que l'application écrit une valeur dans le flux de message d'octets, celle-ci est assemblée immédiatement après la valeur précédemment écrite. XMS gère un curseur interne de manière à mémoriser la position du dernier octet assemblé.

Lorsque l'application envoie un message, le corps de ce message passe en lecture seule. Dans ce mode, l'application peut envoyer le message de manière répétée.

Lorsqu'une application reçoit un message d'octets, le corps de ce message est en lecture seule. L'application peut utiliser les méthodes de lecture appropriées de l'interface IBytesMessage pour lire le contenu du flux de message d'octets. L'application lit les octets en séquence, et XMS gère un curseur interne pour mémoriser la position du dernier octet lu.

Si une application appelle la méthode Reset de l'interface IBytesMessage lorsque le corps d'un message d'octets est accessible en écriture, celui-ci passe en lecture seule. La méthode repositionne également le curseur au début du flux de message d'octets.

Si une application appelle la méthode `Clear Body` de l'interface IMessage pour .NET lorsque le corps d'un message d'octets est en lecture seule, celui-ci devient accessible en écriture. La méthode efface également le corps.

Référence associée

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Messages d'objet
Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

Messages de flux
Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Messages texte
Le corps d'un message texte contient une chaîne.

IBytesMessage (pour l'interface .NET)
Un message d'octets est un message dont le corps contient un flux d'octets.

Messages de mappe

Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Dans chaque paire nom-valeur, le nom est une chaîne qui identifie la valeur, et la valeur est un élément de données d'application qui a l'un des types de données XMS listés dans Tableau 21, à la page 77. L'ordre des paires nom-valeur n'est pas défini. La classe MapMessage contient les méthodes qui permettent de définir et d'extraire les paires nom-valeur.

Une application peut accéder de manière aléatoire à une paire nom-valeur en spécifiant son nom.

Une application .NET peut utiliser la propriété MapNames pour extraire une énumération des noms du corps d'un message de mappe.

Lorsqu'une application extrait la valeur d'une paire nom-valeur, celle-ci peut être convertie par XMS dans un autre type de données. Par exemple, pour obtenir un entier à partir du corps d'un message de mappe, une application peut appeler la méthode GetString de la classe MapMessage, qui retourne l'entier sous forme de chaîne. Les conversions prises en charge sont identiques à celles admises lorsque XMS convertit une valeur de propriété d'un type de données à un autre. Pour plus d'informations sur les conversions prises en charge, voir «Conversion implicite d'une valeur de propriété d'un type de données à un autre», à la page 41.

Lorsqu'un message de mappe a été créé par une application, le corps du message est accessible en lecture et en écriture. Il le reste après l'envoi du message par l'application. Lorsqu'une application reçoit un message de mappe, le corps de ce message est en lecture seule. Si une application appelle la méthode Clear Body de la classe Message alors que le corps du message texte est en lecture seule, celui-ci devient accessible en lecture et en écriture. La méthode efface également le corps.

Concepts associés

Conversion implicite d'une valeur de propriété d'un type de données à un autre
Lorsqu'une application extrait la valeur d'une propriété, celle-ci peut être convertie par XMS dans un autre type de données. De nombreuses règles régissent les conversions prises en charge et la manière dont XMS les réalise.

Référence associée

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Messages d'octets
Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

Messages d'objet
Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

Messages de flux
Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Messages texte
Le corps d'un message texte contient une chaîne.

IMapMessage (pour l'interface .NET)
Un message de mappe est un message dont le corps contient un ensemble de paires nom-valeur où un type de données est associé à chaque valeur.

Messages d'objet

Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

Une application XMS peut recevoir un message d'objet, modifier ses zones d'en-tête et ses propriétés, puis l'envoyer à une autre destination. Une application peut également copier le corps d'un message d'objet et l'utiliser pour former un autre message d'objet. XMS traite le corps d'un message d'objet en tant que tableau d'octets.

Lorsqu'un message d'objet a été créé par une application, le corps du message est accessible en lecture et en écriture. Il le reste après l'envoi du message par l'application. Lorsqu'une application reçoit un message d'objet, le corps du message est accessible en lecture seulement. Si une application appelle la méthode `Clear Body` de l'interface IMessage pour .NET lorsque le corps d'un message d'objet est en lecture seule, le corps devient accessible en lecture et en écriture. La méthode efface également le corps.

Référence associée

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Messages d'octets
Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Messages de flux
Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Messages texte
Le corps d'un message texte contient une chaîne.

IObjectMessage (pour l'interface .NET)
Un message d'objet est un message dont le corps comprend un objet Java ou .NET sérialisé.

Messages de flux

Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Les types de données XMS qui peuvent être associés à une valeur sont listés dans Tableau 21, à la page 77.

Lorsqu'une application a créé un message de flux, le corps de ce message est accessible en écriture. L'application assemble les données d'application dans le corps en appelant les méthodes d'écriture appropriées de l'interface IStreamMessage pour .NET. Chaque fois que l'application écrit une valeur dans le flux de message, cette valeur et son type de données sont immédiatement assemblés à la suite de la valeur écrite précédemment par l'application. XMS gère un curseur interne de manière à mémoriser la position de la dernière valeur assemblée.

Lorsque l'application envoie un message, le corps de ce message passe en lecture seule. Dans ce mode, l'application peut envoyer le message plusieurs fois.

Lorsqu'une application reçoit un message de flux, le corps de ce message est en lecture seule. L'application peut utiliser les méthodes de lecture appropriées de l'interface IStreamMessage pour .NET afin de lire le contenu du flux de messages. L'application lit les valeurs en séquence, et XMS gère un curseur interne pour mémoriser la position de la dernière valeur lue.

Lorsqu'une application lit une valeur à partir du flux de message, celle-ci est convertie par XMS dans un autre type de données. Par exemple, pour lire un entier à partir du flux de messages, une application peut faire appel à la méthode ReadString, qui renvoie l'entier sous forme de chaîne. Les conversions prises en charge sont identiques à celles admises lorsque XMS convertit une valeur de propriété d'un type de données à un autre. Pour plus d'informations sur les conversions prises en charge, voir «Conversion implicite d'une valeur de propriété d'un type de données à un autre», à la page 41.

Si une erreur se produit alors qu'une application est en train de lire une valeur du flux de messages, la position du curseur n'est pas modifiée. L'application peut tenter de lire la valeur sous la forme d'un autre type de données.

Si une application appelle la méthode Reset de l'interface IStreamMessage pour XMS lorsque le corps d'un message de flux est en écriture seule, celui-ci passe en lecture seule. La méthode repositionne également le curseur au début du flux de messages.

Si une application appelle la méthode `Clear Body` de l'interface IMessage pour XMS lorsque le corps d'un message de flux est en lecture seule, le corps devient en écriture seule. La méthode efface également le corps.

Concepts associés

Conversion implicite d'une valeur de propriété d'un type de données à un autre
Lorsqu'une application extrait la valeur d'une propriété, celle-ci peut être convertie par XMS dans un autre type de données. De nombreuses règles régissent les conversions prises en charge et la manière dont XMS les réalise.

Référence associée

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Messages d'octets
Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Messages d'objet
Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

Messages texte
Le corps d'un message texte contient une chaîne.

IStreamMessage (pour l'interface .NET)
Un message de flux est un message dont le corps contient un flux de valeurs et où chaque valeur a un type de données associé. Le contenu du corps est écrit et lu séquentiellement.

Messages texte

Le corps d'un message texte contient une chaîne.

Lorsqu'un message texte a été créé par une application, le corps du message est accessible en lecture et en écriture. Il le reste après l'envoi du message par l'application. Lorsqu'une application reçoit un message texte, le corps du message est accessible en lecture seulement. Si une application appelle la méthode Clear Body de l'interface IMessage pour .NET alors que le corps du message texte est en lecture seule, celui-ci devient accessible en lecture et en écriture. La méthode efface également le corps.

Référence associée

Types de données d'éléments de données d'application
Pour qu'une application XMS puisse échanger des messages avec une application IBM MQ classes for JMS , les deux applications doivent pouvoir interpréter les données d'application dans le corps d'un message de la même manière.

Messages d'octets
Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

Messages d'objet
Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

Messages de flux

Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

ITextMessage (pour l'interface .NET)
Un message texte est un message dont le corps contient une chaîne.

Sélecteurs de message

Une application XMS utilise des sélecteurs de message pour sélectionner les messages qu'elle souhaite recevoir.

Lorsqu'une application crée un consommateur de message, elle peut lui associer une expression de sélecteur de message. Cette expression spécifie les critères de sélection.

Lorsqu'une application se connecte à gestionnaire de files d'attente IBM WebSphere MQ 7.0, la sélection de message s'effectue côté gestionnaire de files d'attente. XMS ne procède à aucune sélection et distribue simplement les messages qu'il reçoit du gestionnaire de files d'attente, fournissant ainsi de meilleures performances.

Une application peut créer plusieurs consommateurs de message avec, pour chacun d'entre eux, sa propre expression de sélecteur de message. Si un message entrant répond aux critères de sélection de plusieurs consommateurs de message, XMS distribue ce message à chacun de ces consommateurs.

Une expression de sélecteur de message peut faire référence aux propriétés d'un message suivantes :

- Propriétés définies par JMS
- Propriétés définie par IBM
- Propriétés définies par l'application

Elle peut également faire référence aux zones d'en-tête de message suivantes :

- JMSCorrelationID
- JMSDeliveryMode
- JMSMessageID
- JMSPriority
- JMSTimestamp
- JMSType

Toutefois, une expression de sélecteur de message ne peut pas faire référence aux données du corps d'un message.

Exemple d'expression de sélecteur de message :

```
JMSPriority > 3 AND manufacturer = 'Jaguar' AND model in ('xj6','xj12')
```

XMS livre un message à un consommateur de message avec cette expression uniquement si le message a une priorité supérieure à 3, une propriété définie par l'application, manufacturer, avec la valeur `Jaguar;` et une autre propriété définie par l'application, model, avec la valeur `xj6` ou `xj12.`

Les règles de syntaxe qui régissent la formation d'une expression de sélecteur de message de XMS sont identiques à celles de IBM MQ classes for JMS. Pour des informations sur la construction d'une expression de sélecteur de message, voir la documentation du produit IBM MQ. Notez que, dans une expression de sélecteur de message, les noms des propriétés définies par JMS doivent être des noms JMS, et les noms des propriétés définies par IBM doivent être des noms IBM MQ classes for JMS. Vous ne pouvez pas utiliser les noms XMS dans une expression de sélecteur de message.

Référence associée

Parties d'un message XMS
Un message XMS est composé d'un en-tête, d'un ensemble de propriétés et d'un corps.

Zones d'en-tête dans un message XMS

Pour permettre à une application XMS d'échanger des messages avec une application WebSphere JMS, l'en-tête d'un message XMS contient les zones d'en-tête du message JMS.

Propriétés d'un message XMS
XMS prend en charge trois sortes de propriétés de message : les propriétés définies par JMS, les propriétés définies par IBM et les propriétés définies par l'application.

Corps d'un message XMS
Le corps d'un message contient des données d'application. Un message peut toutefois ne pas comporter de corps, mais uniquement des zones d'en-tête et des propriétés.

Mappage de messages XMS en messages IBM MQ
Les zones d'en-tête et les propriétés JMS d'un message XMS sont mappées vers des zones dans les structures d'en-tête d'un message IBM MQ.

Mappage de messages XMS en messages IBM MQ

Les zones d'en-tête et les propriétés JMS d'un message XMS sont mappées vers des zones dans les structures d'en-tête d'un message IBM MQ.

Lorsqu'une application XMS est connectée à un gestionnaire de files d'attente IBM MQ, les messages envoyés au gestionnaire de files d'attente sont mappés vers des messages IBM MQ de la même manière que les messages IBM MQ classes for JMS sont mappés vers des messages IBM MQ dans des circonstances similaires.

Si la propriété XMSC_WMQ_TARGET_CLIENT d'un objet Destination est définie sur XMSC_WMQ_TARGET_DEST_JMS, les zones d'en-tête et les propriétés JMS d'un message envoyées à la destination sont mappées vers les zones dans la structure d'en-tête MQMD et MQRFH2 du message IBM MQ. Cette définition de la propriété XMSC_WMQ_TARGET_CLIENT suppose que l'application qui réceptionne le message puisse gérer un en-tête MQRFH2. L'application de réception peut donc être une autre application XMS, une application IBM MQ classes for JMS ou une application IBM MQ native, conçue pour gérer un en-tête MQRFH2.

Si la propriété XMSC_WMQ_TARGET_CLIENT d'un objet Destination est définie sur XMSC_WMQ_TARGET_DEST_MQ, les zones d'en-tête et les propriétés JMS d'un message envoyé à la destination sont mappées vers des zones dans la structure d'en-tête MQMD du message IBM MQ. Le message ne contient pas d'en-tête MQRFH2 ; les zones d'en-tête JMS et les propriétés qui ne peuvent pas être mappées en zones dans la structure d'en-tête MQMD sont ignorées. L'application qui reçoit le message peut donc être une application IBM MQ native qui n'est pas conçue pour gérer un en-tête MQRFH2.

Les messages IBM MQ reçus en provenance d'un gestionnaire de files d'attente sont mappés en messages XMS de la même manière que des messages IBM MQ sont mappés en messages IBM MQ classes for JMS dans des circonstances similaires.

Si un message IBM MQ entrant contient un en-tête MQRFH2, le message XMS résultant contient un corps dont le type est déterminé par la valeur de la propriété **Msd** contenue dans le dossier `mcd` de l'en-tête MQRFH2. Si la propriété **Msd** ne figure pas dans l'en-tête MQRFH2, ou si le message IBM MQ n'a pas d'en-tête MQRFH2, le message XMS résultant a un corps dont le type est déterminé par la valeur de la zone *`Format`* de l'en-tête MQMD. Si la zone *`Format`* est définie à MQFMT_STRING, le message XMS est un message texte. Sinon, le message XMS est un message d'octets. Si le message IBM MQ n'a pas d'en-tête MQRFH2, seules les zones d'en-tête et les propriétés JMS pouvant dériver des zones de l'en-tête MQMD sont définies.

Pour plus d'informations sur le mappage de messages IBM MQ classes for JMS à des messages IBM MQ, voir la documentation du produit IBM MQ.

Référence associée

Parties d'un message XMS
Un message XMS est composé d'un en-tête, d'un ensemble de propriétés et d'un corps.

Zones d'en-tête dans un message XMS

Pour permettre à une application XMS d'échanger des messages avec une application WebSphere JMS, l'en-tête d'un message XMS contient les zones d'en-tête du message JMS.

Propriétés d'un message XMS
XMS prend en charge trois sortes de propriétés de message : les propriétés définies par JMS, les propriétés définies par IBM et les propriétés définies par l'application.

Corps d'un message XMS
Le corps d'un message contient des données d'application. Un message peut toutefois ne pas comporter de corps, mais uniquement des zones d'en-tête et des propriétés.

Sélecteurs de message
Une application XMS utilise des sélecteurs de message pour sélectionner les messages qu'elle souhaite recevoir.

Lecture et écriture du descripteur de message à partir d'une application IBM Message Service Client for .NET

Vous pouvez accéder à toutes les zones du descripteur de message (MQMD) d'un message IBM MQ, à l'exception de StrucId et de Version ; BackoutCount et accessible en lecture, mais pas en écriture. Cette fonction est disponible uniquement lors de la connexion à un gestionnaire de files d'attente IBM WebSphere MQ 6.0 ou version ultérieure et est contrôlée par les propriétés de destination décrites ultérieurement.

Les attributs de message fournis par IBM Message Service Client for .NET facilite la définition des zones MQMD par les applications XMS, ainsi que le pilotage des applications IBM WebSphere MQ.

Certaines restrictions s'appliquent lors de l'utilisation de la messagerie de publication/abonnement. Par exemple, les zones MQMD telles que MsgID et CorrelId sont ignorées si elles sont définies.

La fonction décrite dans cette rubrique n'est pas disponible pour la messagerie de publication / abonnement lorsque vous vous connectez à un gestionnaire de files d'attente IBM WebSphere MQ 6.0 . Elle est également non disponible lorsque la propriété **PROVIDERVERSION** est définie sur 6.

Accès aux données de message IBM MQ depuis une application IBM Message Service Client for .NET

Vous pouvez accéder aux données de message IBM MQ, y compris à l'en-tête MQRFH2 le cas échéant, et à tout autre en-tête IBM MQ existant dans une application IBM Message Service Client for .NET en tant que corps d'un objet JMSBytesMessage.

La fonction décrite dans cette rubrique est disponible uniquement lors de la connexion à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 ou version ultérieure et le fournisseur de messagerie WebSphere MQ est en mode normal.

Les propriétés de l'objet destination déterminent la manière dont l'application XMS accède à l'ensemble d'un message IBM MQ (y compris l'en-tête MQRFH2, le cas échéant) en tant que corps d'un objet JMSBytesMessage.

Identification et résolution des problèmes

Cette section contient des informations permettant de détecter et de résoudre les problèmes rencontrés lors de l'utilisation d'IBM Message Service Client for .NET.

Cette section contient les rubriques suivantes :

- «Configuration de trace pour des applications .NET», à la page 85
- «Configuration FFDC pour des applications .NET», à la page 90
- «Conseils pour le traitement des incidents», à la page 90

Configuration de trace pour des applications .NET

Pour des applications XMS .NET, vous pouvez configurer la trace à partir d'un fichier de configuration de l'application ou de variables d'environnement XMS. Vous pouvez sélectionner les composants à tracer. En général, la trace est utilisée sur les conseils du support IBM.

La fonction de trace pour XMS .NET est basée sur l'infrastructure de trace .NET standard.

Par défaut, tout le traçage est désactivé à l'exception du traçage d'erreur. Vous pouvez activer le traçage et en configurer les paramètres selon l'une des méthodes suivantes :

- En utilisant un fichier de configuration d'application avec un nom correspondant à celui du programme exécutable auquel il est associé, suivi du suffixe `.config`. Par exemple, le fichier de configuration d'application de text.exe se nommerait text.exe.config. L'utilisation d'un fichier de configuration d'application est la meilleure manière d'activer la fonction de trace pour les applications XMS .NET. Pour plus de détails, voir «Configuration de la trace à l'aide d'un fichier de configuration d'application», à la page 86.
- En utilisant les variables d'environnement XMS, par exemple pour les applications XMS C ou C++. Pour plus de détails, voir «Configuration de trace à l'aide de variables d'environnement XMS», à la page 88.

Le format du nom du fichier de trace actif est xms_trace*PID*.log où *PID* représente l'ID de processus de l'application. Par défaut, la taille du fichier de trace actif est limité à 20 Mo. Lorsque la limite est atteinte, le fichier est renommé et archivé. Le format des noms des fichiers archivés est xms_trace*PID_AA.MM.JJ_HH.MM.SS*.log

Par défaut, le nombre de fichiers de trace conservés est 4, soit 1 fichier actif et 3 fichiers archivés. Ces quatre fichiers sont utilisés comme un tampon de roulement jusqu'à ce que l'application s'arrête, le fichier le plus ancien étant supprimé et remplacé par le plus récent. Vous pouvez modifier le nombre de fichiers de trace en spécifiant une valeur différente dans le fichier de configuration d'application. Toutefois, vous devez conserver au minimum 2 fichiers (un fichier actif et un fichier archivé).

Deux formats de fichier de trace sont disponibles :

- Les fichiers de format de base, accessibles en lecture, dans un format WebSphere Application Server. Il s'agit du format de fichier de traçage par défaut. Le format de base n'est pas compatible avec les analyseurs de trace.
- Les fichiers de format avancé, compatibles avec les analyseurs de trace. Vous devez spécifier que vous voulez générer des fichiers de traçage au format avancé dans le fichier de configuration d'application.

Les entrées de trace contiennent les informations suivantes :

- La date et l'heure auxquelles la trace a été consignée
- Le nom de la classe
- Le type de trace
- Le message de trace

L'exemple suivant présente un extrait de trace :

```
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    >    Allocate    Entry
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    >    Initialize    Entry
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    <    Initialize    Exit
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    <    Allocate    Exit
```

Dans cet exemple, le format est :

```
[Date Time:Microsecs]         Thread-id    Classname        Trace-type    Methodname    Entry
or Exit
```

où `Trace-type` correspond à :

> pour Entry
< pour Exit

d pour Debug information

Configuration de la trace à l'aide d'un fichier de configuration d'application

La meilleure manière de configurer une trace pour des applications XMS .NET consiste à utiliser un fichier de configuration d'application. La section de trace de ce fichier contient des paramètres qui permettent de définir les éléments à tracer, l'emplacement du fichier de trace et sa taille maximale, le nombre de fichiers de trace utilisés et le format du fichier de trace.

Pour activer la fonction de trace à l'aide du fichier de configuration de l'application, il suffit de placer le fichier dans le même répertoire que le fichier exécutable de votre application.

La trace peut être activée à la fois par composant et par type de trace. Il est également possible d'activer la trace pour l'intégralité d'un groupe de trace. Vous pouvez activer la fonction de trace pour les composants dans une hiérarchie, individuellement ou collectivement. Les types de trace disponibles incluent :

- Trace de débogage
- Trace d'exception
- Avertissements, messages d'information et messages d'erreur
- Trace d'entrée et de sortie de méthode

L'exemple suivant présente les paramètres de trace définis dans la section Trace d'un fichier de configuration de l'application :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <configSections>
        <sectionGroup name="IBM.XMS">
            <section name="Trace"
                     type="System.Configuration.SingleTagSectionHandler"/>
        </sectionGroup>
    </configSections>

    <IBM.XMS>
        <Trace traceSpecification="*=all=enabled" traceFilePath=""
               traceFileSize="20000000" traceFileNumber="3"
               traceFormat="advanced"/>
    </IBM.XMS>
 </configuration>
```

Le Tableau 22, à la page 87 décrit les paramètres de manière plus détaillée.

Tableau 22. Paramètres de trace du fichier de configuration d'application

Paramètre	**Description**
traceSpecification=*ComponentName*=*type*=*state*	*ComponentName* est le nom de la classe que vous souhaitez tracer. Vous pouvez utiliser un caractère générique * dans ce nom. Par exemple, `*=all=enabled` spécifie que vous voulez tracer toutes les classes, et `IBM.XMS.impl.*=all=enabled` que vous voulez tracer uniquement l'API. *type* peut être l'un des types de trace suivants : • Tous • débogage • événement • EntryExit *state* peut être activé ou désactivé. Vous pouvez chaîner plusieurs éléments de trace ensemble en utilisant le délimiteur ":" (deux-points).
traceFilePath="*filename*"	Si vous ne spécifiez pas le paramètre traceFilePath, ou si traceFilePath est présent mais contient une chaîne vide, le fichier de trace est placé dans le répertoire en cours. Pour stocker le fichier de trace dans un répertoire nommé, spécifiez le nom du répertoire dans traceFilePath, par exemple : `traceFilePath="c:\somepath"`
traceFileSize="*size*"	Taille maximale allouée au fichier de trace. Lorsqu'un fichier atteint cette taille, il est archivé et renommé. La valeur maximale par défaut est 20 Ko, spécifiée sous la forme `traceFileSize="20000000"`.
traceFileNumber="*number*"	Nombre de fichiers de trace à conserver. La valeur par défaut est 4 (1 fichier actif et 3 fichiers archive). La valeur minimale est 2.
traceFormat="*format*"	Le format de trace par défaut est basic. Les fichiers de trace sont produits dans ce format si vous spécifiez `traceFormat="basic"`, si vous ne spécifiez pas de paramètre traceFormat, ou si le paramètre traceFormat est présent mais contient une chaîne vide. Si vous voulez une trace compatible avec des analyseurs de trace, vous devez spécifier `traceFormat="advanced"`.

Les paramètres de trace d'un fichier de configuration d'application sont dynamiques ; ils sont également relus chaque fois que le fichier est sauvegardé ou remplacé. Si des erreurs sont détectées dans le fichier lorsque celui-ci est édité, les valeurs par défaut des paramètres de fichier de trace sont réactivés.

Concepts associés

Configuration de trace à l'aide de variables d'environnement XMS

Activer la trace à l'aide des variables d'environnement XMS est une alternative à l'utilisation d'un fichier de configuration d'application. Ces variables d'environnement sont utilisées uniquement en cas d'absence de spécification de trace dans le fichier de configuration d'application.

Configuration de trace à l'aide de variables d'environnement XMS

Activer la trace à l'aide des variables d'environnement XMS est une alternative à l'utilisation d'un fichier de configuration d'application. Ces variables d'environnement sont utilisées uniquement en cas d'absence de spécification de trace dans le fichier de configuration d'application.

Pour configurer la trace pour une application XMS .NET, définissez les variables d'environnement suivantes avant d'exécuter l'application :

Tableau 23. Paramètres de variable d'environnement pour une trace .NET

Variables d'environnement	**Par défaut**	**Paramètres**	**Explication**
XMS_TRACE_ON	Non applicable	Non applicable : la valeur de cette variable est ignorée	Si XMS_TRACE_ON est défini, tous les composants de trace sont activés par défaut.

Tableau 23. Paramètres de variable d'environnement pour une trace .NET (suite)

Variables d'environnement	Par défaut	Paramètres	Explication
XMS_TRACE_FILE_PATH	Répertoire de travail en cours	/chemin_répertoire/	Chemin de répertoire où sont écrits les enregistrements de trace et d'outil de diagnostic de premier niveau. XMS crée les fichiers de trace et d'outil de diagnostic de premier niveau dans le répertoire de travail en cours, sauf si vous spécifiez un autre emplacement. Vous pouvez spécifier un autre emplacement en définissant la variable d'environnement XMS_TRACE_FILE_PATH avec le nom de chemin d'accès du répertoire dans lequel vous voulez que XMS crée les fichiers de trace et d'outil de diagnostic de premier niveau. Vous devez définir cette variable d'environnement avant de démarrer l'application que vous souhaitez tracer. Assurez-vous que l'ID utilisateur sous lequel l'application s'exécute dispose des droits en écriture sur le répertoire où XMS crée les fichiers de trace et d'outil de diagnostic de premier niveau.
XMS_TRACE_FORMAT	BASIC	BASIC, ADVANCED	Spécifie le format de trace requise, qui peut être BASIC ou ADVANCED. Le format par défaut est BASIC. Le format ADVANCED est compatible avec les outils d'analyse de trace.

Tableau 23. Paramètres de variable d'environnement pour une trace .NET (suite)			
Variables d'environnement	**Par défaut**	**Paramètres**	**Explication**
XMS_TRACE_SPECIFICATION	Non applicable	Voir «Configuration de la trace à l'aide d'un fichier de configuration d'application», à la page 86.	Se substitue à la spécification de trace, qui suit le format indiqué dans «Configuration de la trace à l'aide d'un fichier de configuration d'application», à la page 86.

Concepts associés

Configuration de la trace à l'aide d'un fichier de configuration d'application

La meilleure manière de configurer une trace pour des applications XMS .NET consiste à utiliser un fichier de configuration d'application. La section de trace de ce fichier contient des paramètres qui permettent de définir les éléments à tracer, l'emplacement du fichier de trace et sa taille maximale, le nombre de fichiers de trace utilisés et le format du fichier de trace.

Configuration FFDC pour des applications .NET

Pour l'implémentation .NET de XMS, un fichier est produit pour chaque outil de diagnostic de premier niveau.

Les fichiers FFDC sont stockés dans des fichiers texte lisibles par l'utilisateur. Ces fichiers ont des noms au format `xmsffdc`*`processID_Date`*`T`*`Timestamp`*`.txt`. Exemple de nom de fichier : `xmsffdc264_2006.01.06T13.18.52.990955.txt`. L'horodatage contient une résolution en microsecondes.

Les fichiers commencent avec la date et l'heure à laquelle l'exception s'est produite, suivi du type d'exception. Ils incluent un ID sonde court unique, qui peut être utilisé pour localiser l'emplacement où cette condition FFDC s'est produite.

Aucune configuration n'est nécessaire pour activer la fonction FFDC. Par défaut, tous le fichiers FFDC sont écrits dans le répertoire en cours. Toutefois, si nécessaire, vous pouvez spécifier un répertoire différent en modifiant la valeur de ffdcDirectory dans la section Trace du fichier de configuration d'application. Dans l'exemple suivant, tous les fichiers de trace sont consignés dans le répertoire c:client\ffdc :

```
<IBM.XMS>
      <Trace ffdc=true ffdcDirectory="c:\client\ffdc"/>
</IBM.XMS>
```

Vous pouvez désactiver la fonction de trace en définissant FFDC à false dans la section Trace du fichier de configuration d'application.

Si vous n'utilisez pas de fichier de configuration d'application, la fonction FFDC est activée et la fonction de trace est désactivée.

Conseils pour le traitement des incidents

Vous trouverez ci-après quelques conseils qui vous aideront lors du traitement des incidents liés à XMS.

Une application XMS ne peut pas se connecter à un gestionnaire de files d'attente (MQRC_NOT_AUTHORIZED)

Le comportement du client XMS .NET peut être différent de celui du client IBM MQ JMS . Il est donc possible que l'application XMS ne puisse pas se connecter au gestionnaire de files d'attente alors que l'application JMS le peut.

- Une solution simple à ce problème consiste à essayer d'utiliser un ID utilisateur de moins de 12 caractères et disposant des droits complets dans la liste des droits du gestionnaire de files d'attente. Cette solution n'est pas idéale ; une approche différente mais plus complexe consisterait à utiliser des exits de sécurité. Si vous avez besoin d'une aide supplémentaire pour ce problème, contactez le support IBM.
- Si vous définissez la propriété XMSC_USERID de la fabrique de connexions, celle-ci doit correspondre à l'ID utilisateur et au mot de passe de l'utilisateur connecté. Si vous ne définissez pas cette propriété, le gestionnaire de files d'attente utilise l'ID utilisateur connecté par défaut.
- L'authentification d'utilisateur pour IBM MQ s'effectue à l'aide des détails de l'utilisateur actuellement connecté et non des informations fournies dans les zones XMSC.USERID et XMSC.PASSWORD. Cela a été conçu ainsi afin de maintenir une cohérence avec IBM MQ. Pour plus d'informations sur l'authentification, reportez-vous à *Authentication Information* dans la documentation en ligne du produit IBM MQ.

Connexion redirigée vers le moteur de messagerie

Lorsque vous vous connectez à un bus d'intégration de services WebSphere Application Server 6.0.2 , toutes les connexions peuvent être redirigées du noeud final du fournisseur d'origine vers le moteur de messagerie choisi par le bus pour cette connexion client. Dans ce cas, il redirige toujours la connexion vers un serveur hôte spécifié par le nom d'hôte plutôt que par une adresse IP. Vous pouvez donc rencontrer des problèmes de connexion si le nom d'hôte ne peut pas être résolu.

Pour vous connecter au bus d'intégration de services WebSphere Application Server 6.0.2 , vous devrez peut-être fournir un mappage entre les noms d'hôte et les adresses IP sur votre machine hôte client. Par exemple, vous pouvez indiquer le mappage dans une table d'hôtes locaux sur votre machine hôte client.

Prise en charge de l'authentification par mot de passe de type telnet

Le protocole RTT (Real Time Transport) de XMS .NET ne prend en charge que l'authentification simple par mot de passe, de type Telnet. Le protocole RTT (Real Time Transport) de XMS .NET ne prend pas en charge la qualité de protection..

Définition de valeurs pour le type de propriété double

Sur une plateforme Windows 64 bits, les méthodes SetDoubleProperty() et GetDoubleProperty() risquent de ne pas fonctionner correctement en cas de définition ou d'extraction de valeurs pour le type de propriété double si ces valeurs sont inférieures à Double.Epsilon.

Par exemple, si vous tentez de définir la valeur 4.9E-324 pour une propriété de type double, les plateformes Windows 64 bits la traitent en tant que 0.0. Ainsi, dans un environnement de messagerie répartie, si JMS ou une autre application définit la valeur d'une propriété double 4.9E-324 sur une machine UNIX ou Windows 32 bits et que XMS .NET s'exécute sur une machine 64 bits, la valeur renvoyée par la propriété GetDouble() est 0.0. Il s'agit d'un problème connu lié à Microsoft .NET Framework 2.0 Framework.

Cas d'erreur pouvant être gérés lors de l'exécution

Les codes retour des appels API sont des cas d'erreur qui peuvent être gérés lors de l'exécution. La manière dont vous gérez ce type d'erreurs varie selon que vous utilisez l'API C ou C++.

Détection des erreurs lors de l'exécution

Si une application appelle une fonction d'API C et que cet appel échoue, une réponse avec un code retour autre que XMS_OK est renvoyée avec un bloc d'erreur XMS contenant des informations complémentaires sur les causes de l'échec.

L'API C++ émet une exception lorsqu'une méthode est utilisée.

Une application utilise un programme d'écoute des exceptions afin d'être notifiée de manière asynchrone d'un problème lié à la connexion. Le programme d'écoute des exceptions est fourni et initialisé avec l'API XMS C ou C++.

Gestion des erreurs lors de l'exécution

Certaines conditions d'erreur sont une indication qu'une ressource n'est pas disponible ; l'action que peut effectuer l'application dépend de la fonction XMS appelée. Par exemple, si une connexion au serveur échoue, l'application peut procéder à de nouvelles tentatives régulièrement jusqu'à ce que la connexion soit établie. Un bloc d'erreur ou une exception XMS peut ne pas contenir suffisamment d'informations pour déterminer l'action à entreprendre ; dans ce cas, il existe souvent une exception ou un bloc d'erreur lié contenant des informations de diagnostic plus spécifiques.

Dans l'API C, toute réponse avec un code retour autre que XMS_OK doit être testée, et tout bloc d'erreur doit être transmis sur l'appel API. L'action à entreprendre dépend généralement de la fonction API utilisée par l'application.

Dans l'API C++, incluez toujours les appels aux méthodes dans un bloc try et, pour intercepter tous les types d'exception XMS, spécifiez la classe Exception dans la construction d'interception.

Le programme d'écoute des exceptions est un chemin de cas d'erreur asynchrone qui peut être démarré à tout moment. Lorsque la fonction de programme d'écoute des exceptions est démarrée sur sa propre unité d'exécution, cela indique généralement une défaillance plus grave qu'un cas d'erreur d'API XMS normal. L'action appropriée peut être entreprise, mais vous devez suivre attentivement les règles du modèle d'unités d'exécution XMS, comme décrit dans «Le modèle d'unités d'exécution», à la page 21.

Concepts associés

Le modèle d'unités d'exécution
Des règles générales déterminent comment une application à unités d'exécution multiples peut utiliser des objets XMS.

Message Service Clients for .NET

Cette section présente des informations sur l'utilisation de Message Service Client for .NET. Elles vous aideront à effectuer les tâches liées à la programmation avec XMS.

.NET interfaces

Cette section documente les interfaces de classe .NET , ainsi que leurs propriétés et leurs méthodes.

Le tableau suivant répertorie toutes les interfaces, définies au sein de l'espace de nom `IBM.XMS`.

Tableau 24. Récapitulatif des interfaces de classe .NET

utilisateur	Description
«IBytesMessage», à la page 95	Un message d'octets est un message dont le corps contient un flux d'octets.
«IConnection», à la page 105	Un objet Connection représente la connexion active de l'application à un serveur de messagerie.
«IConnectionFactory», à la page 108	Une application utilise une fabrique de connexions pour créer une connexion.
«IConnectionMetaData», à la page 110	Un objet ConnectionMetaData fournit des informations sur une connexion.
«IDestination», à la page 111	Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Tableau 24. Récapitulatif des interfaces de classe .NET (suite)	
utilisateur	**Description**
«ExceptionListener», à la page 113	Une application utilise un programme d'écoute des exceptions afin d'être notifiée de manière asynchrone d'un problème lié à la connexion.
«IllegalStateException», à la page 113	XMS émet cette exception si une application appelle une méthode à un moment incorrect ou si XMS n'est pas dans un état approprié pour l'opération demandée.
«InitialContext», à la page 113	Une application utilise un objet InitialContext pour créer des objets à partir de définitions d'objet extraits d'un référentiel des objets gérés.
«InvalidClientIDException», à la page 116	XMS émet cette exception si une application tente de définir un identificateur de client pour une connexion, mais que l'identificateur de client n'est pas valide ou qu'il est déjà utilisé.
«InvalidDestinationException», à la page 116	XMS émet cette exception si une application spécifie une destination non valide.
«InvalidSelectorException», à la page 116	XMS émet cette exception si une application fournit une expression de sélecteur de message dont la syntaxe n'est pas valide.
«IMapMessage», à la page 117	Un message de mappe est un message dont le corps contient un ensemble de paires nom-valeur où un type de données est associé à chaque valeur.
«IMessage», à la page 126	Un objet Message représente un message envoyé ou reçu par une application. IMessage est une superclasse des classes de messages comme IMapMessage.
«IMessageConsumer», à la page 132	Une application utilise un consommateur de message pour la réception des messages envoyés vers une destination.
«MessageEOFException», à la page 135	XMS émet cette exception si XMS rencontre la fin d'un flux de messages d'octets lorsqu'une application lit le corps d'un message d'octets.
«MessageFormatException», à la page 135	XMS émet cette exception si XMS rencontre un message dont le format n'est pas valide.
«IMessageListener (délégation)», à la page 136	Une application utilise un programme d'écoute de message pour recevoir des messages de manière asynchrone.
«MessageNotReadableException», à la page 136	XMS émet cette exception si une application tente de lire le corps d'un message en écriture seule.
«MessageNotWritableException», à la page 136	XMS émet cette exception si une application tente d'écrire dans le corps d'un message en lecture seule.
«IMessageProducer», à la page 137	Une application utilise un expéditeur de message pour envoyer des messages vers une destination.

Tableau 24. Récapitulatif des interfaces de classe .NET (suite)	
utilisateur	**Description**
«IObjectMessage», à la page 142	Un message d'objet est un message dont le corps comprend un objet Java ou .NET sérialisé.
«IPropertyContext», à la page 143	IPropertyContext est une superclasse abstraite qui contient des méthodes permettant d'obtenir et de définir des propriétés. Ces méthodes sont héritées par d'autres classes.
«IQueueBrowser», à la page 152	Une application utilise un navigateur de files d'attente pour parcourir les messages sur une file d'attente sans les supprimer.
«Demandeur», à la page 154	Une application utilise un demandeur pour envoyer un message de demande, puis attendre et recevoir la réponse.
«ResourceAllocationException», à la page 156	XMS émet cette exception si XMS ne peut pas allouer les ressources requises par une méthode.
«SecurityException», à la page 156	XMS émet cette exception si l'ID utilisateur et le mot de passe fournis pour authentifier une application sont rejetés. XMS émet également cette exception si une vérification des droits échoue et empêche l'aboutissement d'une méthode.
«ISession», à la page 156	Une session est un contexte à unité d'exécution unique pour l'envoi et la réception de messages.
«IStreamMessage», à la page 167	Un message de flux est un message dont le corps contient un flux de valeurs et où chaque valeur a un type de données associé.
«ITextMessage», à la page 177	Un message texte est un message dont le corps contient une chaîne.
«TransactionInProgressException», à la page 178	XMS émet cette exception si une application demande une opération qui n'est pas valide car une transaction est en cours.
«TransactionRolledBackException», à la page 178	XMS émet cette exception si une application appelle Session.commit() pour valider la transaction en cours, mais que la transaction est ensuite annulée.
XMSC	Pour .NET , les noms et les valeurs de propriété XMS sont définis dans cette classe en tant que constantes publiques. Pour plus de détails, voir «Propriétés des objets XMS», à la page 181.

Tableau 24. Récapitulatif des interfaces de classe .NET (suite)	
utilisateur	**Description**
«XMSException», à la page 178	Si XMS détecte une erreur lors du traitement d'un appel à une méthode .NET , XMS émet une exception. Une exception est un objet qui encapsule des informations relatives à l'erreur. Il existe différents types d'exception XMS et un objet XMSException n'est qu'un seul type d'exception. Toutefois, la classe XMSException est une superclasse des classes d'exception XMS. XMS émet un objet XMSException si aucun autre type d'exception n'est approprié.
«XMSFactoryFactory», à la page 179	Si une application n'utilise pas les objets gérés, utilisez cette classe pour créer des fabriques de connexions, des files d'attente et des rubriques.

La définition de chaque méthode liste les codes d'exception que XMS peut retourner lorsqu'il détecte une erreur au cours du traitement d'un appel à la méthode. Chaque code d'exception est représenté par sa constante nommée, à laquelle correspond une exception.

Concepts associés

Génération de vos propres applications
Vous pouvez générer vos propres applications comme vous générez les modèles d'application.

Ecriture d'applications XMS
Les rubriques de cette section contiennent des informations d'aide à l'écriture d'applications XMS.

Ecriture d'applications XMS .NET
Les rubriques de cette section fournissent des informations qui peuvent vous être utiles si vous écrivez des applications XMS .NET.

Référence associée

Propriétés des objets XMS
Cette chapitre présente les propriétés d'objet définies par XMS.

IBytesMessage

Un message d'octets est un message dont le corps contient un flux d'octets.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IBytesMessage
```

Référence associée

Messages d'octets
Le corps d'un message d'octets contient un flux d'octets. Le corps contient uniquement les données réelles ; il est de la responsabilité de l'application émettrice et de l'application réceptrice d'interpréter ces données.

.NET propriétés

BodyLength - Extraction de la longueur du corps

Interface :

```
Int64 BodyLength
    {
      get;
    }
```

Extrait la longueur du corps du message en octets lorsque le corps du message est en lecture seule.

La valeur renvoyée correspond à la longueur de l'ensemble du corps, quelle que soit la position du curseur pour la lecture du message.

Exceptions :

- XMSException
- MessageNotReadableException

Des méthodes

ReadBoolean - Lecture de valeur booléenne

Interface :

```
Boolean ReadBoolean();
```

Lit une valeur booléenne depuis le flux de message d'octets.

Paramètres :
Aucun

Retour :
Valeur booléenne lue.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte - Lecture d'octet

Interface :

```
Int16   ReadSignedByte();
```

Lit l'octet suivant du flux de message d'octets sous la forme d'un entier signé de 8 bits.

Paramètres :
Aucun

Retour :
Octet lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Lecture d'octets

Interface :

```
Int32   ReadBytes(Byte[] array);
Int32   ReadBytes(Byte[] array, Int32 length);
```

Lit un tableau d'octets du flux de message d'octets, en commençant à la position du curseur.

Paramètres :

array (sortie)
Mémoire tampon pour contenir le tableau d'octets lu. Si le nombre d'octets restant à lire à partir du flux avant l'appel est supérieur ou égal à la longueur de la mémoire tampon, celle-ci est remplie. Dans le cas contraire, elle est remplie partiellement avec les octets restants.

Si vous spécifiez un pointeur Null en entrée, la méthode ignore les octets sans les lire. Si le nombre d'octets restant à lire à partir du flux avant l'appel est supérieur ou égal à la longueur de la mémoire tampon, le nombre d'otets ignorés est égale à la longueur de la mémoire tampon. Sinon, les octets restants sont ignorés. Le curseur reste sur la position suivante pour lire le flux de message d'octets.

length (entrée)
Longueur de la mémoire tampon en octets

Retour :
Nombre d'octets lus dans la mémoire tampon. Si la mémoire tampon est partiellement remplie, la valeur est inférieure à la longueur de la mémoire tampon, indiquant qu'il ne reste plus d'octets à lire. S'il ne reste plus d'octets à lire à partir du flux avant l'appel, la valeur est `XMSC_END_OF_STREAM`.

Si vous spécifiez un pointeur Null en entrée, la méthode ne renvoie aucune valeur.

Exceptions :

- XMSException
- MessageNotReadableException

ReadChar - Lecture de caractère

Interface :

```
Char    ReadChar();
```

Lit les 2 octets suivants à partir du flux de message d'octets sous la forme d'un caractère.

Paramètres :
Aucun

Retour :
Caractère lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Lecture d'un nombre en virgule flottante à double précision

Interface :

```
Double  ReadDouble();
```

Lit les 8 octets suivants à partir du flux de message d'octets sous la forme d'un nombre en virgule flottante à double précision.

Paramètres :
Aucun

Retour :
Le nombre en virgule flottante à double précision lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Lecture du nombre en virgule flottante

Interface :

```
Single  ReadFloat();
```

Lit les 4 octets suivants du flux de message d'octets sous la forme d'un nombre en virgule flottante.

Paramètres :
Aucun

Retour :
Le nombre en virgule flottante lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Lecture d'un entier

Interface :

```
Int32   ReadInt();
```

Lit les 4 octets suivants du flux de message d'octets sous la forme d'un entier signé de 32 bits.

Paramètres :
Aucun

Retour :
Entier lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Lecture d'un entier long

Interface :

```
Int64   ReadLong();
```

Lit les 8 octets suivants du flux de message d'octets sous la forme d'un entier signé de 64 bits.

Paramètres :
Aucun

Retour :
Entier long lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadShort - Lecture d'un entier court

Interface :

```
Int16   ReadShort();
```

Lit les 2 octets suivants du flux de message d'octets sous la forme d'un entier signé de 16 bits.

Paramètres :
Aucun

Retour :
Entier court lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Lecture d'octet non signé

Interface :

```
Byte    ReadByte();
```

Lit l'octet suivant du flux de message d'octets sous la forme d'un entier signé de 8 bits.

Paramètres :
Aucun

Retour :
Octet lu.

Exceptions :

- XMSException
- MessageNotReadableException

- MessageEOFException

ReadUnsignedShort - Lecture d'un entier court non signé

Interface :

```
Int32   ReadUnsignedShort();
```

Lit les 2 octets suivants du flux de message d'octets sous la forme d'un entier non signé de 16 bits.

Paramètres :
Aucun

Retour :
Entier court non signé lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUTF - Lecture de chaîne UTF

Interface :

```
String  ReadUTF();
```

Lit une chaîne codée en UTF-8 à partir du flux de message d'octets.

Remarque : Avant d'appeler ReadUTF(), assurez-vous que le curseur de la mémoire tampon pointe sur le début du flux de message d'octets.

Paramètres :
Aucun

Retour :
Objet String encapsulant la chaîne lue.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - Réinitialisation

Interface :

```
void Reset();
```

Met le corps du message en mode lecture seule et repositionne le curseur au début du flux de message d'octets.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException
- MessageNotReadableException

WriteBoolean - Ecriture d'une valeur booléenne

Interface :

```
void  WriteBoolean(Boolean value);
```

Ecrit une valeur booléenne sur le flux de message d'octets.

Paramètres :

value (entrée)
Valeur booléenne à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteByte - Ecriture d'un octet

Interface :

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Ecrit un octet sur le flux de message d'octets.

Paramètres :

value (entrée)
Octet à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteBytes - Ecriture d'octets

Interface :

```
void  WriteBytes(Byte[] value);
```

Ecrit un tableau d'octets sur le flux de message d'octets.

Paramètres :

value (entrée)
Tableau des octets à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteBytes - Ecriture d'un tableau partiel d'octets

Interface :

```
void  WriteBytes(Byte[] value, int offset, int length);
```

Ecrit un tableau partiel d'octets sur le flux de message d'octets, comme défini par la longueur spécifiée.

Paramètres :

value (entrée)
Tableau des octets à écrire.

offset (entrée)
Point de départ du tableau des octets à écrire.

length (entrée)
Nombre d'octets à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteChar - Ecriture d'un caractère

Interface :

```
void  WriteChar(Char value);
```

Ecrit un caractère sur le flux de message d'octets sous forme de 2 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Caractère à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteDouble - Ecriture d'un nombre en virgule flottante à double précision

Interface :

```
void  WriteDouble(Double value);
```

Convertit un nombre en virgule flottante à double précision en entier long et écrit celui-ci sur le flux de message d'octets sous forme de 8 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Le nombre en virgule flottante à double précision à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteFloat - Ecriture d'un nombre en virgule flottante

Interface :

```
void  WriteFloat(Single value);
```

Convertit un nombre en virgule flottante en entier et écrit cet entier sur le flux de message d'octets sous forme de 4 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Le nombre en virgule flottante à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteInt - Ecriture d'un entier

Interface :

```
void  WriteInt(Int32 value);
```

Ecrit un entier sur le flux de message d'octets sous forme de 4 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Entier à écrire.

Retour :
Vide

Exceptions :

- XMSException

- MessageNotWritableException

WriteLong - Ecriture d'un entier long

Interface :

```
void  WriteLong(Int64 value);
```

Ecrit un entier long sur le flux de message d'octets sous forme de 8 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Entier long à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteObject - Ecriture d'un objet

Interface :

```
void  WriteObject(Object value);
```

Ecrit l'objet spécifié dans le flux de message d'octets.

Paramètres :

value (entrée)
Objet à écrire, qui doit être une référence au type primitif.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteShort - Ecriture d'un entier court

Interface :

```
void  WriteShort(Int16 value);
```

Ecrit un entier court sur le flux de message d'octets sous forme de 2 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Entier court à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteUTF - Ecriture d'une chaîne UTF

Interface :

```
void  WriteUTF(String value);
```

Ecrit une chaîne codée en UTF-8 sur le flux de message d'octets.

Paramètres :

value (entrée)
Objet String encapsulant la chaîne à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

Propriétés et méthodes héritées

Les propriétés suivantes sont héritées de l'interface IMessage :

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Les méthodes suivantes sont héritées de l'interface IMessage :

clearBody, clearProperties, PropertyExists

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnection

Un objet Connection représente la connexion active de l'application à un serveur de messagerie.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnection
```

Pour obtenir la liste des propriétés XMS définies d'un objet Connection, voir «Propriétés de Connection», à la page 182.

.NET propriétés

ClientID - Extraction et définition de l'ID client

Interface :

```
String ClientID
      {
        get;
        set;
      }
```

Obtient et définit l'identificateur de client pour la connexion.

L'identificateur de client peut être préconfiguré par l'administrateur dans une fabrique de connexions ou affectée par la définition de ClientID.

Un identificateur de client est utilisé uniquement pour prendre en charge des abonnements durables dans le domaine publication/abonnement ; il est ignoré dans le domaine point-à-point.

Si une application définit un identificateur de client pour une connexion, elle doit le faire immédiatement après la création de la connexion et avant toute autre opération sur cette connexion. Si l'application tente de définir un identificateur de client à un moment ultérieur, l'appel émet l'exception IllegalStateException.

Cette propriété n'est pas valide pour une connexion en temps réel à un courtier.

Exceptions :

- XMSException
- IllegalStateException
- InvalidClientIDException

ExceptionListener - Extraction et définition d'un programme d'écoute des exceptions

Interface :

```
    ExceptionListener ExceptionListener
        {
          get;
          set;
        }
```

Extrait le programme d'écoute des exceptions enregistré avec la connexion et enregistre un programme d'écoute des exceptions avec la connexion.

Si aucun programme d'écoute des exceptions n'est enregistré avec la connexion, la méthode renvoie la valeur Null. Si un programme d'écoute des exceptions est déjà enregistré avec la connexion, vous pouvez annuler l'enregistrement en spécifiant une valeur Null au lieu du programme d'écoute des exceptions.

Pour plus d'informations sur l'utilisation des programmes d'écoute des exceptions, voir«Message et programme d'écoute des exceptions dans .NET», à la page 49.

Exceptions :

- XMSException

Metadata - Extraction des métadonnées

Interface :

```
IConnectionMetaData MetaData
    {
      get;
    }
```

Extrait les métadonnées pour la connexion.

Exceptions :

- XMSException

Des méthodes

Close - Fermeture de la connexion

Interface :

```
void  Close();
```

Ferme la connexion.

Si une application tente de fermer une connexion déjà fermée, l'appel est ignoré.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

CreateSession - Création de session

Interface :

```
    ISession CreateSession(Boolean transacted,
                           AcknowledgeMode acknowledgeMode);
```

Crée une session.

Paramètres :

transacted (entrée)
La valeur `True` signifie que la session est transactionnelle. La valeur `False` signifie que la session n'est pas transactionnelle.

Pour une connexion en temps réel à un courtier, la valeur doit être `False`.

acknowledgeMode (entrée)
Indique la méthode d'accusé de réception des messages reçus par une application. L'énumérateur AcknowledgeMode doit utiliser l'une des valeurs suivantes :

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Pour une connexion en temps réel à un courtier, la valeur doit être `AcknowledgeMode.AutoAcknowledge` ou `AcknowledgeMode.DupsOkAcknowledge`

Ce paramètre est ignoré si la session est transactionnelle. pour plus d'informations sur les modes d'accusé de réception, voir «Accusé de réception de message», à la page 26.

Retour :
Objet Session

Exceptions :

- XMSException

Start - Démarrage d'une connexion

Interface :

```
void  Start();
```

Démarre ou redémarre la livraison des messages entrant pour la connexion. L'appel est ignoré si la connexion est déjà démarrée.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

Stop - Arrêt de la connexion

Interface :

```
void  Stop();
```

Arrête la livraison des messages entrant pour la connexion. L'appel est ignoré si la connexion est déjà arrêtée.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionFactory

Une application utilise une fabrique de connexions pour créer une connexion.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionFactory
```

Pour obtenir la liste des propriétés XMS définies d'un objet ConnectionFactory, voir «Propriétés de ConnectionFactory», à la page 182.

Concepts associés

Objets ConnectionFactories et Connection
Un objet ConnectionFactory fournit un modèle utilisé par une application pour créer un objet Connection. L'application utilise l'objet Connection pour créer un objet Session.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Connexions sécurisées à un gestionnaire de files d'attente IBM MQ
Pour permettre à une application XMS .NET de créer des connexions sécurisées à un gestionnaire de files d'attente IBM MQ, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Connexions sécurisées à un moteur de messagerie WebSphere Application Server service integration bus
Pour permettre à une application XMS .NET de créer des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Des méthodes

CreateConnection - Création d'une fabrique de connexions (avec ID utilisateur par défaut)

Interface :

```
IConnection CreateConnection();
```

Crée une fabrique de connexions avec les propriétés par défaut.

Si vous vous connectez à WebSphere MQ alors que XMSC_USERID n'est pas défini, le gestionnaire de files d'attente utilise l'ID utilisateur de l'utilisateur connecté par défaut. Si vous avez besoin d'authentifications de niveau connexion pour des utilisateurs individuels, vous pouvez écrire un exit d'authentification de client, configuré dans WebSphere MQ.

Paramètres :
Aucun

Exceptions :

- XMSException

CreateConnection - Création d'une connexion (avec ID utilisateur spécifié)

Interface :

```
IConnection CreateConnection(String userId, String password);
```

Crée une connexion avec une identité d'utilisateur spécifiée.

Si vous vous connectez à WebSphere MQ alors que XMSC_USERID n'est pas défini, le gestionnaire de files d'attente utilise l'ID utilisateur de l'utilisateur connecté par défaut. Si vous avez besoin d'authentifications de niveau connexion pour des utilisateurs individuels, vous pouvez écrire un exit d'authentification de client, configuré dans WebSphere MQ.

La connexion est créée en mode arrêté. Aucun message n'est distribué jusqu'à ce que l'application appelle **`Connection.start()`**.

Paramètres :

userID (entrée)
Objet String encapsulant l'ID utilisateur à utiliser pour authentifier l'application. Si vous indiquez une valeur Null, une tentative de création de connexion sans authentification est effectuée.

password (entrée)
Objet String encapsulant le mot de passe à utiliser pour authentifier l'application. Si vous indiquez une valeur Null, une tentative de création de connexion sans authentification est effectuée.

Retour :
Objet Connection.

Exceptions :

- XMSException
- XMS_X_SECURITY_EXCEPTION

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionMetaData

Un objet ConnectionMetaData fournit des informations sur une connexion.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionMetaData
```

Pour obtenir la liste des propriétés XMS définies d'un objet ConnectionMetaData, voir «Propriétés de ConnectionMetaData», à la page 189.

.NET propriétés

JMSXPropertyNames - Extraction des propriétés de message JMS définies

Interface :

```
System.Collections.IEnumerator JMSXPropertyNames
```

```
    {
      get;
    }
```

Renvoie une énumération des noms des propriétés de message définies par JMS et prises en charge par la connexion.

Les propriétés de message JMS définis ne sont pas prises en charge par une connexion en temps réel à un courtier.

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IDestination

Une destination représente l'endroit où une application envoie des messages, l'endroit d'où elle en reçoit, ou les deux.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IDestination
```

Pour obtenir la liste des propriétés XMS définies d'un objet Destination, voir «Propriétés de Destination», à la page 189.

Concepts associés

Objets ConnectionFactories et Connection
Un objet ConnectionFactory fournit un modèle utilisé par une application pour créer un objet Connection. L'application utilise l'objet Connection pour créer un objet Session.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Destinations
Une application XMS utilise un objet Destination pour spécifier la destination des messages envoyés et la source des messages reçus.

Caractères génériques de destination
XMS prend en charge les caractères génériques de destination et assure leur transmission vers l'endroit où ils sont requis pour une correspondance. Il existe un schéma de caractères génériques différent pour chaque type de serveur pouvant être associé à XMS.

identificateur URI de rubrique
L'identificateur URI (Uniform Resource Identifier) de rubrique spécifie le nom de la rubrique ; il peut également en spécifier une ou plusieurs propriétés.

Identificateurs URI de file d’attente
L'identificateur URI d'une file d'attente indique le nom de cette file d'attente ; il peut également en spécifier une ou plusieurs propriétés.

Destinations temporaires
Les applications XMS peuvent créer et utiliser des destinations temporaires.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

Propriétés requises pour les objets Destination gérés
Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

.NET propriétés

Name - Extraction du nom de destination

Interface :

```
String Name
{
  get;
}
```

Extrait le nom de la destination. Le nom est une chaîne encapsulant le nom d'une file d'attente ou le nom d'une rubrique.

Exceptions :

- XMSException

TypeId - Extraction du type de destination

Interface :

```
DestinationType TypeId
{
  get;
}
```

Extrait le type de la destination. Le type de la destination peut être l'une des valeurs suivantes :

```
DestinationType.Queue
DestinationType.Topic
```

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty,

SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ExceptionListener

Hiérarchie d'héritage :

Aucun

Une application utilise un programme d'écoute des exceptions afin d'être notifiée de manière asynchrone d'un problème lié à la connexion.

Si une application utilise une connexion uniquement pour consommer des messages de manière asynchrone, le programme d'écoute des exceptions constitue le seul moyen pour cette application d'être prévenue des problèmes liés à la connexion. Dans les autres cas, un programme d'écoute des exceptions peut fournir une manière plus directe de prendre connaissance d'un problème de connexion que d'attendre l'appel synchrone suivant à XMS.

Délégation

ExceptionListener – Programme d'écoute des exceptions

Interface :

```
public delegate void ExceptionListener(Exception ex)
```

Notifie l'application d'un problème avec une connexion.

Les méthodes qui implémentent cette délégation peuvent être enregistrées avec la connexion.

Pour plus d'informations sur l'utilisation des programmes d'écoute des exceptions, voir«Message et programme d'écoute des exceptions dans .NET», à la page 49.

Paramètres :

exception (entrée)
Pointeur vers une exception créée par XMS.

Retour :

Vide

IllegalStateException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.Exception
            |
            +----IBM.XMS.IllegalStateException
```

XMS émet cette exception si une application appelle une méthode à un moment incorrect ou si XMS n'est pas dans un état approprié pour l'opération demandée.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

InitialContext

Une application utilise un objet InitialContext pour créer des objets à partir de définitions d'objet extraits d'un référentiel des objets gérés.

Hiérarchie d'héritage :

Aucun

Concepts associés

Propriétés InitialContext

Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS

L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

Extraction d'objets gérés

XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Objets InitialContext

Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

.NET propriétés

Environment - Extraction de l'environnement

Interface :

```
Hashtable Environment
{
     get;
}
```

Extrait l'environnement.

Exceptions :

- Les exceptions sont spécifiques au service d'annuaire en cours d'utilisation.

Constructeurs

InitialContext – Création d'un contexte initial

Interface :

```
InitialContext(Hashtable env);
```

Crée un objet InitialContext.

Paramètres :

Les informations requises pour établir une connexion au référentiel des objets gérés est fourni pour le constructeur dans un environnement Hashtable.

Exceptions :

- XMSException

Des méthodes

AddToEnvironment - Ajout d'une propriété à l'environnement

Interface :

```
Object AddToEnvironment(String propName, Object propVal);
```

Ajout d'une propriété à l'environnement.

Paramètres :

propName (entrée)
Objet de type String encapsulant le nom de la propriété à ajouter.

propVal (entrée)
Valeur de la propriété à ajouter.

Retour :
Ancienne valeur de la propriété.

Exceptions :

- Les exceptions sont spécifiques au service d'annuaire en cours d'utilisation.

Close - Fermeture de ce contexte

Interface :

```
void Close()
```

Ferme ce contexte.

Paramètres :
Aucun

Retour :
Aucun

Exceptions :

- Les exceptions sont spécifiques au service d'annuaire en cours d'utilisation.

Lookup – Recherche d'un objet dans un contexte initial

Interface :

```
Object Lookup(String name);
```

Crée un objet à partir d'une définition d'objet extraite du référentiel des objets gérés.

Paramètres :

name (entrée)
Objet de type String encapsulant le nom de l'objet géré à extraire. Il peut s'agir d'un nom simple ou complexe. Pour plus de détails, voir «Extraction d'objets gérés», à la page 66.

Retour :
Objet IConnectionFactory ou IDestination, selon le type d'objet extrait. Si la fonction peut accéder au répertoire mais ne trouve pas l'objet requis, une valeur Null est renvoyée.

Exceptions :

- Les exceptions sont spécifiques au service d'annuaire en cours d'utilisation.

RemoveFromEnvironment - Suppression d'une propriété de l'environnement

Interface :

```
Object RemoveFromEnvironment(String propName);
```

Supprime une propriété de l'environnement.

Paramètres :

propName (entrée)
Objet de type String encapsulant le nom de la propriété à supprimer.

Retour :
Objet ayant été supprimé.

Exceptions :

- Les exceptions sont spécifiques au service d'annuaire en cours d'utilisation.

InvalidClientIDException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidClientIDException
```

XMS émet cette exception si une application tente de définir un identificateur de client pour une connexion, mais que l'identificateur de client n'est pas valide ou qu'il est déjà utilisé.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

InvalidDestinationException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidDestinationException
```

XMS émet cette exception si une application spécifie une destination non valide.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

InvalidSelectorException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
```

```
          |
          +----IBM.XMS.InvalidSelectorException
```

XMS émet cette exception si une application fournit une expression de sélecteur de message dont la syntaxe n'est pas valide.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

IMapMessage

Un message de mappe est un message dont le corps contient un ensemble de paires nom-valeur où un type de données est associé à chaque valeur.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IMapMessage
```

Lorsqu'une application obtient la valeur d'une paire nom-valeur, cette valeur peut être convertie par XMS en un autre type de données. Pour plus d'informations sur cette forme de conversion implicite, voir «Messages de mappe», à la page 78.

Référence associée

Messages de mappe
Le corps d'un message de mappe contient un ensemble de paires nom-valeur, où chaque valeur est associée à un type de données.

.NET propriétés

MapNames - Extraction de noms de mappe

Interface :

```
System.Collections.IEnumerator MapNames
{
  get;
}
```

Extrait une énumération des noms présents dans le corps du message de mappe.

Exceptions :

- XMSException

Des méthodes

GetBoolean - Extraction d'une valeur booléenne

Interface :

```
Boolean GetBoolean(String name);
```

Extrait la valeur booléenne identifiée par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
un objet String encapsulant le nom qui identifie la valeur booléenne.

Retour :
La valeur booléenne extraite du corps du message de mappe.

Exceptions :

- XMSException

GetByte - Extraction d'un octet

Interface :

```
Byte    GetByte(String name);
    Int16   GetSignedByte(String name);
```

Extraction de l'octet identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Un objet String encapsulant le nom qui identifie l'octet.

Retour :
Octet extrait à partir du corps du message de mappe. Aucune conversion de données n'est effectuée sur l'octet.

Exceptions :

- XMSException

GetBytes - Extraction d'octets

Interface :

```
Byte[]  GetBytes(String name);
```

Extrait le tableau d'octets identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie le tableau d'octets.

Retour :
Nombre d'octets dans le tableau.

Exceptions :

- XMSException

GetChar - Extraction d'un caractère

Interface :

```
Char    GetChar(String name);
```

Extrait le caractère identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie le caractère.

Retour :
Caractère extrait à partir du corps du message de mappe.

Exceptions :

- XMSException

GetDouble - Extraction du nombre en virgule flottante à double précision

Interface :

```
Double  GetDouble(String name);
```

Extrait le nombre en virgule flottante à double précision identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie le nombre en virgule flottante à double précision.

Retour :
Nombre en virgule flottante à double précision extrait du corps du message de mappe.

Exceptions :

- XMSException

GetFloat - Extraction d'un nombre en virgule flottante

Interface :

```
Single  GetFloat(String name);
```

Extrait le nombre en virgule flottante identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie le nombre en virgule flottante.

Retour :
Nombre en virgule flottante extrait du corps du message de mappe.

Exceptions :

- XMSException

GetInt - Extraction d'un entier

Interface :

```
Int32   GetInt(String name);
```

Extrait l'entier identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie l'entier.

Retour :
Entier extrait à partir du corps du message de mappe.

Exceptions :

- XMSException

GetLong - Extraction d'un entier long

Interface :

```
Int64   GetLong(String name);
```

Extrait l'entier long identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie l'entier long.

Retour :
Entier long extrait à partir du corps du message de mappe.

Exceptions :

- XMSException

GetObject - Extraction d'un objet

Interface :

```
Object  GetObject(String name);
```

Extrait une référence à la valeur d'une paire nom-valeur, à partir du corps du message de mappe. La paire nom-valeur est identifiée par nom.

Paramètres :

name (entrée)
Objet String encapsulant le nom de la paire nom-valeur.

Retour :
La valeur, dont le type d'objet peut être :

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Exceptions :
XMSException

GetShort - Extrait un entier court

Interface :

```
Int16   GetShort(String name);
```

Extrait l'entier court identifié par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie l'entier court.

Retour :
Entier court extrait du corps du message de mappe.

Exceptions :

- XMSException

GetString - Extraction d'une chaîne

Interface :

```
String  GetString(String name);
```

Extrait la chaîne identifiée par nom à partir du corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie la chaîne dans le corps du message de mappe.

Retour :
Objet String encapsulant la chaîne extraite du corps du message de mappe. Si une conversion de données est requise, cette valeur correspond à la chaîne après conversion.

Exceptions :

- XMSException

ItemExists - Vérification de l'existence d'une paire nom-valeur

Interface :

```
Boolean ItemExists(String name);
```

Vérifie si le corps du message de mappe contient une paire nom-valeur du nom spécifié.

Paramètres :

name (entrée)
Objet String encapsulant le nom de la paire nom-valeur.

Retour :

- `True` si le corps du message de mappe contient une paire nom-valeur du nom spécifié.
- `False` si le corps du message de mappe ne contient pas de paire nom-valeur du nom spécifié.

Exceptions :

- XMSException

SetBoolean - Définition d'une valeur booléenne

Interface :

```
void  SetBoolean(String name, Boolean value);
```

Définit une valeur booléenne dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom qui identifie la valeur booléenne dans le corps du message de mappe.

value (entrée)
Valeur booléenne à définir.

Retour :
Vide

Exceptions :

- XMSException

SetByte - Définition d'un octet

Interface :

```
void  SetByte(String name, Byte value);
void  SetSignedByte(String name, Int16 value);
```

Définit un octet dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant l'octet dans le corps du message de mappe.

value (entrée)
Octet à définir.

Retour :
Vide

Exceptions :

- XMSException

SetBytes - Définition d'octets

Interface :

```
void  SetBytes(String name, Byte[] value);
```

Définit un tableau d'octets dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant le tableau d'octets dans le corps du message de mappe.

value (entrée)
Tableau d'octets à définir.

Retour :
Vide

Exceptions :

- XMSException

SetChar - Définition d'un caractère

Interface :

```
void  SetChar(String name, Char value);
```

Définit un caractère sur 2 octets dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant le caractère dans le corps du message de mappe.

value (entrée)
Caractère à définir.

Retour :
Vide

Exceptions :

- XMSException

SetDouble - Définition du nombre en virgule flottante à double précision

Interface :

```
void  SetDouble(String name, Double value);
```

Définit un nombre en virgule flottante à double précision dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant le nombre en virgule flottante à double précision dans le corps du message de mappe.

value (entrée)
Le nombre en virgule flottante à double précision à définir.

Retour :
Vide

Exceptions :

- XMSException

SetFloat - Définition d'un nombre en virgule flottante

Interface :

```
void  SetFloat(String name, Single value);
```

Définit un nombre en virgule flottante dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant le nombre en virgule flottante dans le corps du message de mappe.

value (entrée)
Nombre en virgule flottante à définir.

Retour :
Vide

Exceptions :

- XMSException

SetInt - Définition d'un entier

Interface :

```
void  SetInt(String name, Int32 value);
```

Définit un entier dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant l'entier dans le corps du message de mappe.

value (entrée)
Entier à définir.

Retour :
Vide

Exceptions :

- XMSException

SetLong - Définition d'un entier long

Interface :

```
void  SetLong(String name, Int64 value);
```

Définit un entier long dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant l'entier long dans le corps du message de mappe.

value (entrée)
Entier long à définir.

Retour :
Vide

Exceptions :

- XMSException

SetObject - Définition d'un objet

Interface :

```
void  SetObject(String name, Object value);
```

Définit une valeur, qui doit être un type primitif XMS, dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant la valeur dans le corps du message de mappe.

value (entrée)
Tableau d'octets contenant la valeur à définir.

Retour :
Vide

Exceptions :

- XMSException

SetShort - Définition d'un entier court

Interface :

```
void  SetShort(String name, Int16 value);
```

Définit un entier court dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant l'entier court dans le corps du message de mappe.

value (entrée)
Entier court à définir.

Retour :
Vide

Exceptions :

- XMSException

SetString - Définition d'une chaîne

Interface :

```
void  SetString(String name, String value);
```

Définit une chaîne dans le corps du message de mappe.

Paramètres :

name (entrée)
Objet String encapsulant le nom identifiant la chaîne dans le corps du message de mappe.

value (entrée)
Objet de type String encapsulant la chaîne à définir.

Retour :
Vide

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les propriétés suivantes sont héritées de l'interface IMessage :

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Les méthodes suivantes sont héritées de l'interface IMessage :

clearBody, clearProperties, PropertyExists

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessage

Un objet Message représente un message envoyé ou reçu par une application. IMessage est une superclasse des classes de messages comme IMapMessage.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

Pour obtenir la liste des zones d'en-tête de message JMS pour un objet Message, voir «Zones d'en-tête dans un message XMS», à la page 71. Pour obtenir la liste des propriétés JMS définies pour un objet Message, voir «Propriétés de message définies par JMS», à la page 73. Pour obtenir la liste des propriétés IBM définies pour un objet Message, voir «Propriétés de message définies par IBM», à la page 74. Pour obtenir la liste des propriétés JMS_IBM_MQMD* pour un objet Message, voir «Propriétés JMS_IBM_MQMD*», à la page 194

Les messages sont supprimés par le récupérateur de place. Lorsqu'un message est supprimé, les ressources qu'il utilisait sont libérées.

Référence associée

Messages XMS

Cette section décrit la structure et le contenu des messages XMS et explique comment les applications traitent les messages XMS.

.NET propriétés

GetJMSCorrelationID - Extraction et définition de JMSCorrelationID

Interface :

```
String JMSCorrelationID
{
  get;
  set;
}
```

Extrait et définit l'identificateur de corrélation du message en tant qu'objet String.

Exceptions :

- XMSException

JMSDeliveryMode - Extraction et définition de JMSDeliveryMode

Interface :

```
DeliveryMode JMSDeliveryMode
{
  get;
  set;
}
```

Extraction et définition du mode de livraison du message.

Le mode de livraison du message peut prendre l'une des valeurs suivantes :

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Pour un nouveau message qui n'a pas été envoyé, le mode de livraison est `DeliveryMode.Persistent`, sauf dans le cas d'une connexion en temps réel à un courtier pour lequel le mode de livraison est `DeliveryMode.NonPersistent`. Pour un message reçu, la méthode renvoie le mode de livraison qui a été défini par l'appel IMessageProducer.send() lorsque le message a été envoyé, sauf si l'application de réception change le mode de livraison avec JMSDeliveryMode.

Exceptions :

- XMSException

JMSDestination - Extraction et définition de JMSDestination

Interface :

```
IDestination JMSDestination
{
  get;
  set;
}
```

Extrait et définit la destination du message.

La destination est définie par l'appel IMessageProducer.send() lors de l'envoi du message. La valeur de JMSDestination est ignorée. Toutefois, vous pouvez utiliser JMSDestination pour changer la destination d'un message reçu.

Pour un nouveau message qui n'a pas été envoyé, la méthode renvoie un objet Destination Null, sauf si l'application émettrice définit une destination avec JMSDestination. Pour un message reçu, la méthode renvoie un objet Destination pour la destination qui a été définie par l'appel IMessageProducer.send() lorsque le message a été envoyé, sauf si l'application de réception change la destination avec JMSDestination.

Exceptions :

- XMSException

JMSExpiration - Extraction et définition de JMSExpiration

Interface :

```
Int64 JMSExpiration
{
  get;
  set;
}
```

Extrait et définit le délai d'expiration du message.

Le délai d'expiration est défini par l'appel IMessageProducer.send() lors de l'envoi du message. Sa valeur est calculée en ajoutant la durée de vie, telle que spécifiée par l'application émettrice, à l'heure d'envoi du message. Le délai d'expiration est exprimé en millisecondes, depuis le 1er janvier 1970, 00:00:00 GMT.

Pour un nouveau message qui n'a pas été envoyé, le délai d'expiration est de 0, sauf si l'application émettrice définit un délai d'expiration différent avec JMSExpiration. Pour un message reçu, la méthode renvoie le délai d'expiration qui a été défini par l'appel IMessageProducer.send() lors de l'envoi du message, sauf si l'application de réception change le délai d'expiration avec JMSExpiration.

Si la durée de vie est égale à 0, l'appel IMessageProducer.send() définit le délai d'expiration à 0 pour indiquer que le message n'expire pas.

XMS supprime les messages expirés et ne les distribue pas aux applications.

Exceptions :

- XMSException

JMSMessageID - Extraction et définition de JMSMessageID

Interface :

```
String JMSMessageID
{
  get;
  set;
}
```

Extrait et définit l'identificateur du message en tant qu'objet String encapsulant l'identificateur de message.

L'identificateur de message est défini par l'appel IMessageProducer.send() lors de l'envoi du message. Pour un message reçu, la méthode renvoie l'identificateur de message qui a été défini par l'appel IMessageProducer.send() lors de l'envoi du message, sauf si l'application de réception change l'identificateur de message avec JMSMessageID.

Si le message ne contient pas d'identificateur de message, la méthode renvoie une valeur Null.

Exceptions :

- XMSException

JMSPriority - Extraction et définition JMSPriority

Interface :

```
Int32 JMSPriority
{
  get;
  set;
}
```

Extrait et définit la priorité du message.

La priorité est définie par l'appel IMessageProducer.send() lors de l'envoi du message. La valeur est un entier compris entre 0, priorité la plus faible, et 9, priorité la plus forte.

Pour un nouveau message qui n'a pas été envoyé, la priorité est 4 sauf si l'application émettrice définit une priorité différente avec JMSPriority. Pour un message reçu, la méthode renvoie la priorité définie par l'appel IMessageProducer.send() lors de l'envoi du message, sauf si l'application de réception change la priorité avec JMSPriority.

Exceptions :

- XMSException

JMSRedelivered - Extraction et définition de JMSRedelivered

Interface :

```
Boolean JMSRedelivered
{
  get;
  set;
}
```

Obtient une indication relative à la distribution du message et indique si le message a été redistribué. L'indication est définie par l'appel IMessageConsumer.receive() lors de la réception du message.

Cette propriété a les valeurs suivantes :

- `True` si le message est redistribué.
- `False` si le message n'est pas redistribué.

Pour une connexion en temps réel à un courtier, la valeur est toujours `False`.

Une indication de redistribution définie par JMSRedelivered avant l'envoi du message est ignorée par l'appel IMessageProducer.send() lors de l'envoi du message ; elle est également ignorée et remplacée par l'appel IMessageConsumer.receive() lors de la réception du message. Toutefois, vous pouvez utiliser JMSRedelivered pour changer l'indication pour un message reçu.

Exceptions :

- XMSException

JMSReplyTo - Extraction et définition de JMSReplyTo

Interface :

```
IDestination JMSReplyTo
{
  get;
  set;
}
```

Extrait et définit la destination où une réponse au message doit être envoyée.

La valeur de cette propriété est un objet Destination pour la destination où une réponse doit être envoyée. Un objet Destination Null signifie qu'aucune réponse n'est attendue.

Exceptions :

- XMSException

JMSTimestamp - Extraction et définition de JMSTimestamp

Interface :

```
Int64 JMSTimestamp
{
  get;
  set;
}
```

Extrait et définit l'heure à laquelle le message a été envoyé.

L'horodatage est défini par l'appel IMessageProducer.send() lorsque le message est envoyé ; il est exprimé en millisecondes, à partir du 1er janvier 1970 00:00:00 GMT.

Pour un nouveau message qui n'a pas été envoyé, l'horodatage est 0, sauf si l'application émettrice définit un horodatage différent avec JMSTimestamp. Pour un message reçu, la méthode renvoie l'horodatage défini par l'appel IMessageProducer.send() lors de l'envoi du message, sauf si l'application de réception change l'horodatage avec JMSTimestamp.

Exceptions :

- XMSException

Remarques :

1. Si l'horodatage n'est pas défini, la méthode retourne 0, mais n'émet pas d'exception.

JMSType - Extraction et définition de JMSType

Interface :

```
String JMSType
{
  get;
  set;
}
```

Extrait et définit le type du message.

La valeur de JMSType est une chaîne encapsulant le type du message. Si une conversion de données est requise, cette valeur correspond au type après conversion.

Exceptions :

- XMSException

PropertyNames - Extraction des propriétés

Interface :

```
System.Collections.IEnumerator PropertyNames
{
  get;
}
```

Extrait une énumération des noms de propriétés du message.

Exceptions :

- XMSException

Des méthodes

Acknowledge - Accusé de réception

Interface :

```
void  Acknowledge();
```

Accuse réception de ce message et de tous les messages sans accusé de réception précédents reçus par la session.

Une application peut appeler cette méthode si le mode d'accusé de réception de la session est AcknowledgeMode.ClientAcknowledge. Les appels à la méthode sont ignorés si la session est associée à un autre mode d'accusé de réception ou s'il s'agit d'une session transactionnelle.

Les messages reçus mais pour lesquels aucun accusé de réception n'a été émis peuvent être redistribués.

Pour plus d'information sur l'accusé de réception des messages, voir «Accusé de réception de message», à la page 26.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException
- IllegalStateException

ClearBody - Suppression du corps

Interface :

```
void  ClearBody();
```

Supprime le corps du message. Les zones d'en-tête et les propriétés du message ne sont pas supprimées.

Si une application supprime le corps d'un message, le corps est conservé sous forme de corps vide dans un nouveau message. L'état d'un corps vide dans un nouveau message dépend du type de corps du message. Pour plus d'informations, voir «Corps d'un message XMS», à la page 76.

Une application peut supprimer un corps de message à tout moment, quel que soit son état. Si un corps de message est en lecture seule, l'application doit commencer par le supprimer pour l'écrire ensuite.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

ClearProperties - Suppression des propriétés

Interface :

```
void  ClearProperties();
```

Supprime les propriétés du message. Les zones d'en-tête et le corps du message ne sont pas supprimés.

Si une application supprime les propriétés d'un message, les propriétés deviennent lisibles et accessibles en écriture.

Une application peut supprimer les propriétés d'un message à tout moment, quel que soit leur état. Si les propriétés du message sont en lecture seule, l'application doit commencer par les supprimer pour les réécrire ensuite.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

PropertyExists - Vérification de l'existence d'une propriété

Interface :

```
Boolean PropertyExists(String propertyName);
```

Vérifie si le message a une propriété avec le nom spécifié.

Paramètres :

propertyName (entrée)
Objet String encapsulant le nom de la propriété.

Retour :

- `True` si le message a une propriété du nom spécifié.
- `False` si le message n'a pas de propriété avec le nom spécifié.

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessageConsumer

Une application utilise un consommateur de message pour la réception des messages envoyés vers une destination.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageConsumer
```

Pour obtenir la liste des propriétés XMS définies d'un objet MessageConsumer, voir «Propriétés de MessageConsumer», à la page 198.

.NET propriétés

MessageListener - Extraction et définition du programme d'écoute de message

Interface :

```
MessageListener MessageListener
{
  get;
  set;
}
```

Extrait le programme d'écoute de message enregistré avec le consommateur de message, puis enregistre un programme d'écoute de message avec le consommateur de message.

Si aucun programme d'écoute de message n'est enregistré avec le consommateur de message, MessageListener est Null. Si un programme d'écoute de message est déjà enregistré avec le consommateur de message, vous pouvez annuler l'enregistrement en spécifiant une valeur null à la place.

Pour plus d'informations sur l'utilisation des programmes d'écoute de message, voir «Message et programme d'écoute des exceptions dans .NET», à la page 49.

Exceptions :

- XMSException

MessageSelector - Extraction du sélecteur de message

Interface :

```
String MessageSelector
{
  get;
}
```

Extrait le sélecteur de message du consommateur de message. La valeur de retour est un objet de type String encapsulant l'expression de sélecteur de message. Si une conversion de données est requise, cette valeur est l'expression de sélecteur de message après la conversion. Si le consommateur de message n'a pas de sélecteur de message, la valeur de MessageSelector est un objet Null de type String.

Exceptions :

- XMSException

Des méthodes

Close - Fermeture de consommateur de message

Interface :

```
void  Close();
```

Ferme le consommateur de message.

Si une application tente de fermer un consommateur de message qui est déjà fermé, l'appel est ignoré.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

Receive - Réception

Interface :

```
IMessage Receive();
```

Recevez le message suivant pour le consommateur de message. L'appel attend un message sans limite de temps ou jusqu'à ce que le consommateur de message soit fermé.

Paramètres :
Aucun

Retour :
Pointeur vers l'objet Message. Si le consommateur de message est fermé pendant que l'appel attend un message, la méthode renvoie un pointeur vers un objet Message de valeur Null.

Exceptions :

- XMSException

Receive - Réception (avec un intervalle d'attente)

Interface :

```
IMessage Receive(Int64 delay);
```

Recevez le message suivant pour le consommateur de message. L'appel attend un message pendant une période de temps spécifiée ou jusqu'à la fermeture du consommateur de message.

Paramètres :

temps d'attente (entrée)
Temps, en millisecondes, pendant lequel l'appel attend un message. Si vous spécifiez un intervalle d'attente égal à 0, l'appel attend un message sans limite de temps.

Retour :
Pointeur vers l'objet Message. Si aucun message n'arrive au cours de l'intervalle d'attente, ou si le consommateur de message est fermé alors que l'appel est en attente d'un message, la méthode renvoie un pointeur vers un objet Message de valeur Null, mais n'émet pas d'exception.

Exceptions :

- XMSException

ReceiveNoWait - Réception sans attente

Interface :

```
IMessage ReceiveNoWait();
```

Réception du message suivant pour le consommateur de message, si un message est disponible immédiatement.

Paramètres :
Aucun

Retour :
Pointeur vers un objet Message. Si aucun message n'est disponible immédiatement, la méthode renvoie un pointeur vers un objet de message Null.

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

MessageEOFException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
            |
            +----IBM.XMS.MessageEOFException
```

XMS émet cette exception si XMS rencontre la fin d'un flux de messages d'octets lorsqu'une application lit le corps d'un message d'octets.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

MessageFormatException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
            |
            +----IBM.XMS.MessageFormatException
```

XMS émet cette exception si XMS rencontre un message dont le format n'est pas valide.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

IMessageListener (délégation)

Hiérarchie d'héritage :

Aucun

Une application utilise un programme d'écoute de message pour recevoir des messages de manière asynchrone.

Délégation

MessageListener - Programme d'écoute de message

Interface :

```
public delegate void MessageListener(IMessage msg);
```

Distribue un message à un consommateur de message de manière asynchrone.

Les méthodes qui implémentent cette délégation peuvent être enregistrées avec la connexion.

Pour plus d'informations sur l'utilisation des programmes d'écoute de message, voir «Message et programme d'écoute des exceptions dans .NET», à la page 49.

Paramètres :

mesg (entrée)

Objet Message.

Retour :

Vide

MessageNotReadableException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotReadableException
```

XMS émet cette exception si une application tente de lire le corps d'un message en écriture seule.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

MessageNotWritableException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotWritableException
```

XMS émet cette exception si une application tente d'écrire dans le corps d'un message en lecture seule.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

IMessageProducer

Une application utilise un expéditeur de message pour envoyer des messages vers une destination.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageProducer
```

Pour obtenir la liste des propriétés XMS définies d'un objet MessageProducer, voir «Propriétés de MessageProducer», à la page 198.

.NET propriétés

DeliveryMode - Extraction et définition du mode de livraison par défaut

Interface :

```
DeliveryMode DeliveryMode
{
  get;
  set;
}
```

Extrait et définit le mode de livraison par défaut pour les messages envoyées par l'expéditeur de message.

Le mode de livraison par défaut peut prendre l'une des valeurs suivantes :

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Pour une connexion en temps réel à un courtier, la valeur doit être `DeliveryMode.NonPersistent`.

La valeur par défaut est `DeliveryMode.Persistent`, sauf pour une connexion en temps réelle à un courtier, pour laquelle la valeur par défaut est `DeliveryMode.NonPersistent`.

Exceptions :

- XMSException

Destination - Définition de la destination

Interface :

```
IDestination Destination
{
  get;
}
```

Définit la destination pour l'expéditeur de message.

Paramètres :
Aucun

Retour :
Objet Destination. Si l'expéditeur de message n'a pas de destination, la méthode renvoie un objet Destination Null.

Exceptions :

- XMSException

DisableMsgID - Extraction et définition de la désactivation de l'identificateur d'ID message

Interface :

```
Boolean DisableMessageID
{
  get;
  set;
}
```

Extrait une indication qui détermine si l'application de réception nécessite que des ID de message soient inclus dans les messages envoyés par l'expéditeur de message.

Sur une connexion à un gestionnaire de files d'attente ou sur une connexion en temps réel à un courtier, cet indicateur est ignoré. Sur une connexion à un bus d'intégration de services, l'indicateur est appliqué.

DisabledMsgID peut prendre l'une des valeurs suivantes :

- `True` si l'application de réception ne nécessite pas que des ID message soient inclus dans les messages envoyés par l'expéditeur de message.
- `False` si l'application de réception nécessite que des ID message soient inclus dans les messages envoyés par l'expéditeur de message.

Exceptions :

- XMSException

DisableMsgTS - Extraction et définition de la désactivation de l'indicateur d'horodatage

Interface :

```
Boolean DisableMessageTimestamp
{
  get;
  set;
}
```

Extrait une indication qui détermine si l'application de réception nécessite que des horodatages soient inclus dans les messages envoyés par l'expéditeur de message.

Sur une connexion en temps réel à un courtier, cet indicateur est ignoré. Sur une connexion à un gestionnaire de files d'attente ou sur une connexion à un bus d'intégration de services, l'indicateur est appliqué.

DisableMsgTS peut prendre l'une des valeurs suivantes :

- `True` si l'application de réception ne nécessite pas que les horodatages soient inclus dans les messages envoyés par l'expéditeur de message.
- `False` si l'application de réception nécessite que les horodatages soient inclus dans les messages envoyés par l'expéditeur de message.

Retour :

Exceptions :

- XMSException

Priority - Extraction et définition des priorités par défaut

Interface :

```
Int32 Priority
{
  get;
  set;
}
```

Extrait et définit la priorité par défaut pour les messages envoyés par l'expéditeur de message.

La valeur de la priorité de message par défaut est un entier compris entre 0, priorité la plus basse, et9, priorité la plus élevée.

Sur une connexion en temps réel à un courtier, la priorité d'un message est ignorée.

Exceptions :

- XMSException

TimeToLive - Extraction et définition de la durée de vie par défaut

Interface :

```
Int64 TimeToLive
{
  get;
  set;
}
```

Extrait et définit le temps d'existence d'un message avant son expiration.

Le temps est mesuré à partir du moment où l'expéditeur de message envoie le message et représente la durée de vie par défaut (en millisecondes). Une valeur de 0 signifie qu'un message n'expire jamais.

Pour une connexion en temps réel à un courtier, la valeur est toujours 0.

Exceptions :

- XMSException

Des méthodes

Close - Fermeture de l'expéditeur de message

Interface :

```
void  Close();
```

Ferme l'expéditeur de message.

Si une application tente de fermer un expéditeur de message déjà fermé, l'appel est ignoré.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

Send - Envoi

Interface :

```
void Send(IMessage msg) ;
```

Envoyer un message à la destination spécifiée lors de la création de l'expéditeur de message. Le message est envoyé avec le mode de livraison, la priorité et la durée de vie par défaut.

Paramètres :

msg (entrée)
Objet Message.

Retour :
Vide

Exceptions :

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - Envoi (avec indication du mode de livraison, de la priorité et de la durée de vie)

Interface :

```
void Send(IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive);
```

Envoyer un message à la destination spécifiée lors de la création de l'expéditeur de message. Le message est envoyé avec le mode de livraison, la priorité et la durée de vie par défaut.

Paramètres :

msg (entrée)
Objet Message.

deliveryMode (entrée)
Le mode de livraison du message peut prendre l'une des valeurs suivantes :

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

Pour une connexion en temps réel à un courtier, la valeur doit être `DeliveryMode.NonPersistent`.

priority (entrée)
Priorité du message. La valeur peut être un entier compris entre 0, priorité la plus faible, et 9, priorité la plus forte. Sur une connexion en temps réel à un courtier, la valeur est ignorée.

timeToLive (entrée)
Durée de vie du message (en millisecondes). La valeur 0 indique que le message n'arrive jamais à expiration. Pour une connexion en temps réel à un courtier, la valeur doit être 0.

Retour :
Vide

Exceptions :

- XMSException

- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Send - Envoi (vers une destination spécifiée)

Interface :

```
void Send(IDestination dest, IMessage msg) ;
```

Envoyez un message à une destination spécifiée si vous utilisez un expéditeur de message pour lequel aucune destination n'a été spécifiée lors de la création de l'expéditeur de message. Le message est envoyé avec le mode de livraison, la priorité et la durée de vie par défaut.

Généralement, vous indiquez une destination lors de la création de l'expéditeur de message ; si vous ne le faites pas, vous devez spécifier une destination chaque fois que vous envoyez un message.

Paramètres :

dest (entrée)
Objet Destination.

msg (entrée)
Objet Message.

Retour :
Vide

Exceptions :

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - Envoi (vers une destination spécifiée, avec indication du mode de livraison, de la priorité et de la durée de vie)

Interface :

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

Envoyez un message à une destination spécifiée si vous utilisez un expéditeur de message pour lequel aucune destination n'a été spécifiée lors de la création de l'expéditeur de message. Le message est envoyé avec le mode de livraison, la priorité et la durée de vie par défaut.

Généralement, vous indiquez une destination lors de la création de l'expéditeur de message ; si vous ne le faites pas, vous devez spécifier une destination chaque fois que vous envoyez un message.

Paramètres :

dest (entrée)
Objet Destination.

msg (entrée)
Objet Message.

deliveryMode (entrée)
Le mode de livraison du message peut prendre l'une des valeurs suivantes :

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Pour une connexion en temps réel à un courtier, la valeur doit être `DeliveryMode.NonPersistent`.

priority (entrée)
Priorité du message. La valeur peut être un entier compris entre 0, priorité la plus faible, et 9, priorité la plus forte. Sur une connexion en temps réel à un courtier, la valeur est ignorée.

timeToLive (entrée)
Durée de vie du message (en millisecondes). La valeur 0 indique que le message n'arrive jamais à expiration. Pour une connexion en temps réel à un courtier, la valeur doit être 0.

Retour :
Vide

Exceptions :

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IObjectMessage

Un message d'objet est un message dont le corps comprend un objet Java ou .NET sérialisé.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IObjectMessage
```

Référence associée
Messages d'objet
Le corps d'un message d'objet contient un objet sérialisé Java ou .NET.

.NET propriétés

Objet - Extraction et définition de l'objet sous forme d'octets

Interface :

```
System.Object Object
    {
      get;
      set;
    }
```

```
Byte[] GetObject();
```

Extrait et défini l'objet qui forme le corps du message d'objet.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException
- MessageNotWritableException

Propriétés et méthodes héritées

Les propriétés suivantes sont héritées de l'interface IMessage :

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Les méthodes suivantes sont héritées de l'interface IMessage :

clearBody, clearProperties, PropertyExists

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPropertyContext

IPropertyContext est une superclasse abstraite qui contient des méthodes permettant d'obtenir et de définir des propriétés. Ces méthodes sont héritées par d'autres classes.

Hiérarchie d'héritage :
Aucun

Des méthodes

GetBooleanProperty – Extraction de la propriété booléenne

Interface :

```
Boolean GetBooleanProperty(String property_name);
```

extrait la valeur de la propriété booléenne avec le nom spécifié.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetByteProperty – Extraction de la propriété d'octet

Interface :

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

Extrait la valeur de la propriété d'octet identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetBytesProperty – Extraction de la propriété de tableau d'octets

Interface :

```
Byte[]  GetBytesProperty(String property_name) ;
```

Extrait la valeur du tableau d'octets identifié par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Nombre d'octets dans le tableau.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetCharProperty – Extraction de la propriété de caractère

Interface :

```
Char    GetCharProperty(String property_name) ;
```

Extrait la valeur de la propriété de caractère sur 2 octets identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetDoubleProperty – Extraction de la propriété de nombre en virgule flottante à double précision

Interface :

```
Double  GetDoubleProperty(String property_name) ;
```

Extrait la valeur de la propriété du nombre en virgule flottante à double précision identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetFloatProperty – Extraction de la propriété de virgule flottante

Interface :

```
Single  GetFloatProperty(String property_name) ;
```

Extrait la valeur de la propriété de virgule flottante identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetIntProperty – Extraction de la propriété d'entier

Interface :

```
Int32   GetIntProperty(String property_name) ;
```

Extrait la valeur de la propriété d'entier identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetLongProperty – Extraction de la propriété d'entier long

Interface :

```
Int64   GetLongProperty(String property_name) ;
```

Extrait la valeur de la propriété d'entier long identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetObjectProperty – Extraction de la propriété d'objet

Interface :

```
Object  GetObjectProperty( String property_name) ;
```

Extrait la valeur et le type de données de la propriété identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété, dont le type d'objet peut être l'un des suivants :

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
```

String

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetShortProperty – Extraction de la propriété d'entier court

Interface :

```
Int16   GetShortProperty(String property_name) ;
```

Extrait la valeur de la propriété d'entier court identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Valeur de la propriété.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

GetStringProperty – Extraction de la propriété de chaîne

Interface :

```
String  GetStringProperty(String property_name) ;
```

Extrait la valeur de la chaîne identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

Retour :
Objet String encapsulant la chaîne, c'est-à-dire la valeur de la propriété. Si une conversion de données est requise, cette valeur correspond à la chaîne après conversion.

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException

SetBooleanProperty – Définition de la propriété booléenne

Interface :

```
void SetBooleanProperty( String property_name,  Boolean value) ;
```

Définit la valeur de la propriété booléenne identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetByteProperty – Définition de la propriété d'octet

Interface :

```
void SetByteProperty( String property_name,  Byte value) ;
      void SetSignedByteProperty( String property_name,  Int16 value) ;
```

Définit la valeur de la propriété d'octet identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetBytesProperty – Définition de la propriété de tableau d'octets

Interface :

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

Définit la valeur de la propriété de tableau d'octets identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété qui correspond à un tableau d'octets.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetCharProperty – Définition de la propriété de caractère

Interface :

```
void SetCharProperty( String property_name,  Char value) ;
```

Définit la valeur de la propriété de caractère sur 2 octets identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetDoubleProperty – Définition de la propriété de nombre en virgule flottante à double précision

Interface :

```
void SetDoubleProperty( String property_name,  Double value) ;
```

Définit la valeur de la propriété du nombre en virgule flottante à double précision identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetFloatProperty – Définition de la propriété de virgule flottante

Interface :

```
void SetFloatProperty( String property_name,  Single value) ;
```

Définit la valeur de la propriété de virgule flottante identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetIntProperty – Définition de la propriété d'entier

Interface :

```
void SetIntProperty( String property_name,  Int32 value) ;
```

Définit la valeur de la propriété d'entier identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetLongProperty – Définition de la propriété d'entier long

Interface :

```
void SetLongProperty( String property_name,  Int64 value) ;
```

Définit la valeur de la propriété d'entier long identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetObjectProperty – Définition de la propriété d'objet

Interface :

```
void SetObjectProperty( String property_name,  Object value) ;
```

Définit la valeur et le type de données de la propriété identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

objectType (entrée)
Valeur de la propriété, dont le type d'objet peut être l'un des suivants :

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value (entrée)
Valeur de la propriété qui correspond à un tableau d'octets.

length (entrée)
Nombre d'octets dans le tableau.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetShortProperty – Définition de la propriété d'entier court

Interface :

```
void SetShortProperty( String property_name,  Int16 value) ;
```

Définit la valeur de la propriété d'entier court identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

SetStringProperty – Définition de la propriété de chaîne

Interface :

```
void SetStringProperty( String property_name,  String value);
```

Définit la valeur de la propriété de chaîne identifiée par nom.

Paramètres :

property_name (entrée)
Objet String encapsulant le nom de la propriété.

value (entrée)
Objet String encapsulant la chaîne, c'est-à-dire la valeur de la propriété.

Retour :
Vide

Contexte d'unité d'exécution :
Déterminé par la sous-classe

Exceptions :

- XMSException
- MessageNotWritableException

IQueueBrowser

Une application utilise un navigateur de files d'attente pour parcourir les messages sur une file d'attente sans les supprimer.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
   |
   +----IBM.XMS.IQueueBrowser
```

Propriétés .NET

MessageSelector - Extraction du sélecteur de message

Interface :

```
String MessageSelector
{
  get;
}
```

Extrait le sélecteur de message pour le navigateur de files d'attente.

Le sélecteur de message est un objet de type String encapsulant l'expression de sélecteur de message. Si une conversion de données est requise, cette valeur est l'expression de sélecteur de message après la conversion. Si le navigateur de files d'attente n'a pas de sélecteur de message, la méthode renvoie un objet de type String Null.

Exceptions :

- XMSException

Queue - Extraction de file d'attente

Interface :

```
IDestination Queue
{
  get;
}
```

extrait la file d'attente associée au navigateur de files d'attente en tant qu'objet de destination représentant la file d'attente.

Exceptions :

- XMSException

Des méthodes

Close - Fermeture du navigateur de files d'attente

Interface :

```
void  Close();
```

Ferme le navigateur de files d'attente.

Si une application tente de fermer un navigateur de files d'attente qui est déjà fermé, l'appel est ignoré.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException

GetEnumerator - Extraction de messages

Interface :

```
IEnumerator GetEnumerator();
```

Extrait une liste des messages de la file d'attente.

La méthode renvoie une expression énumérative qui encapsule une liste d'objets Message. L'ordre des objets Message est le même que celui dans lequel les messages sont extraits de la file d'attente. L'application peut alors utiliser l'expression énumérative pour parcourir chaque message à son tour.

L'expression énumérative est mise à jour de manière dynamique à mesure que les messages sont insérés dans la file d'attente et supprimés de la file d'attente. Chaque fois que l'application appelle IEnumerator.MoveNext () pour parcourir le message suivant dans la file d'attente, le message reflète le contenu en cours de la file d'attente.

Si une application appelle cette méthode plus d'une fois pour un navigateur de files d'attente, chaque appel renvoie une nouvelle expression énumérative. L'application peut ainsi utiliser plus d'une expression énumérative pour parcourir les messages dans une file d'attente et gérer plusieurs positions dans la file d'attente.

Paramètres :
Aucun

Retour :
Objet Iterator.

Exceptions :

- XMSException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Demandeur

Une application utilise un demandeur pour envoyer un message de demande, puis attendre et recevoir la réponse.

Hiérarchie d'héritage :
Aucun

Constructeurs

Requestor – Création d'un demandeur

Interface :

```
Requestor(ISession sess, IDestination dest);
```

Crée un demandeur.

Paramètres :

sess (entrée)
Objet Session. La session ne doit pas être transactionnelle et doit être associée à l'un des modes d'accusé de réception suivants :

```
AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.DupsOkAcknowledge
```

dest (entrée)
Objet Destination représentant la destination vers laquelle l'application peut envoyer des messages de demande.

Contexte d'unité d'exécution :
Session associée au demandeur

Exceptions :

- XMSException

Des méthodes

Close – Fermeture du demandeur

Interface :

```
void  Close();
```

Ferme le demandeur.

Si une application tente de fermer un demandeur déjà fermé, l'appel est ignoré.

Remarque : Lorsqu'une application ferme un demandeur, la session associée n'est pas fermée. A cet égard, XMS se comporte différemment de JMS.

Paramètres :
Aucun

Retour :
Vide

Contexte d'unité d'exécution :
Tous

Exceptions :

- XMSException

Request – Réponse à une demande

Interface :

```
IMessage Request(IMessage requestMessage);
```

Envoie un message de demande, puis attend et reçoit une réponse de l'application ayant reçu le message de demande.

Un appel à cette demande bloque jusqu'à la réception d'une réponse ou la fin de session.

Paramètres :

requestMessage (entrée)
Objet Message encapsulant le message de demande.

Retour :
Pointeur vers l'objet Message encapsulant le message de demande.

Contexte d'unité d'exécution :
Session associée au demandeur

Exceptions :

- XMSException

ResourceAllocationException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.ResourceAllocationException
```

XMS émet cette exception si XMS ne peut pas allouer les ressources requises par une méthode.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

SecurityException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.SecurityException
```

XMS émet cette exception si l'ID utilisateur et le mot de passe fournis pour authentifier une application sont rejetés. XMS émet également cette exception si une vérification des droits échoue et empêche l'aboutissement d'une méthode.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

ISession

Une session est un contexte à unité d'exécution unique pour l'envoi et la réception de messages.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.ISession
```

Pour obtenir la liste des propriétés XMS définies d'un objet Session, voir «Propriétés de Session», à la page 198.

Propriétés .NET

AcknowledgeMode - Extraction du mode d'accusé de réception

Interface :

```
AcknowledgeMode AcknowledgeMode
    {
      get;
    }
```

Extrait le mode d'accusé de réception pour la session.

Le mode d'accusé de réception est spécifié lors de la création de la session.

Si la session n'est pas transactionnelle, le mode d'accusé de réception correspond à l'une des valeurs suivantes :

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

pour plus d'informations sur les modes d'accusé de réception, voir «Accusé de réception de message», à la page 26.

Une session transactionnelle n'a pas de mode d'accusé de réception. Si la session est transactionnelle, la méthode renvoie la valeur `AcknowledgeMode.SessionTransacted`.

Exceptions :

- XMSException

Transacted - Détermine s'il s'agit d'une session transactionnelle

Interface :

```
Boolean Transacted
    {
      get;
    }
```

Détermine si la session est transactionnelle.

L'état transactionnel peut être :

- `True` pour une session transactionnelle.
- `False` pour une session non transactionnelle.

Pour une connexion en temps réel à un courtier, la méthode renvoie toujours `False`.

Exceptions :

- XMSException

Des méthodes

Close - Fermeture de session

Interface :

```
void Close();
```

Ferme la session. Si la session est transactionnelle, toute transaction en cours est annulée.

Si une application tente de fermer une session déjà fermée, l'appel est ignoré.

Paramètres :
Aucun

Retour :
Vide

Contexte d'unité d'exécution :
Tous

Exceptions :

- XMSException

Commit - Validation

Interface :

```
void Commit();
```

Valide tous les messages traités lors de la transaction en cours.

La session doit être transactionnelle.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException
- IllegalStateException
- TransactionRolledBackException

CreateBrowser - Création d'un navigateur de files d'attente

Interface :

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

Crée un navigateur de files d'attente pour la file d'attente spécifiée.

Paramètres :

queue (entrée)
Objet Destination représentant la file d'attente.

Retour :
Objet QueueBrowser.

Exceptions :

- XMSException
- InvalidDestinationException

CreateBrowser - Création d'un navigateur de files d'attente (avec sélecteur de message)

Interface :

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

Création d'un navigateur de files d'attente pour la file d'attente spécifiée avec un sélecteur de message.

Paramètres :

queue (entrée)
Objet Destination représentant la file d'attente.

selector (entrée)
Objet String encapsulant une expression de sélecteur de message. Seuls les messages dont les propriétés correspondent à l'expression du sélecteur de message sont distribués sur le navigateur de files d'attente.

Un objet String Null signifie qu'aucun sélecteur de message n'est défini pour le navigateur de files d'attente.

Retour :
Objet QueueBrowser.

Exceptions :

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateBytesMessage - Création d'un message d'octets

Interface :

```
IBytesMessage CreateBytesMessage();
```

Crée un message d'octets.

Paramètres :
Aucun

Retour :
Objet BytesMessage.

Exceptions :

- XMSException
- IllegalStateException (The session is closed)

CreateConsumer - Création d'un consommateur

Interface :

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

Crée un consommateur de message pour la destination spécifiée.

Paramètres :

dest (entrée)
Objet Destination.

Retour :
Objet MessageConsumer.

Exceptions :

- XMSException
- InvalidDestinationException

CreateConsumer - Création d'un consommateur (avec sélecteur de message)

Interface :

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector) ;
```

Crée un consommateur de message pour la destination spécifiée avec un sélecteur de message.

Paramètres :

dest (entrée)
Objet Destination.

selector (entrée)
Objet String encapsulant une expression de sélecteur de message. Seuls les messages dont les propriétés correspondent à l'expression du sélecteur de message sont distribués sur le consommateur de message.

Un objet String Null signifie qu'aucun sélecteur de message n'est défini pour le consommateur de message.

Retour :
Objet MessageConsumer.

Exceptions :

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateConsumer - Création d'un consommateur (avec sélecteur de message et indicateur de message local)

Interface :

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector,
                                Boolean noLocal) ;
```

Crée un consommateur de message pour la destination spécifiée avec un sélecteur de message et, si la destination est une rubrique, spécifie si le consommateur de message reçoit les messages publiés par sa propre connexion.

Paramètres :

dest (entrée)
Objet Destination.

selector (entrée)
Objet String encapsulant une expression de sélecteur de message. Seuls les messages dont les propriétés correspondent à l'expression du sélecteur de message sont distribués sur le consommateur de message.

Un objet String Null signifie qu'aucun sélecteur de message n'est défini pour le consommateur de message.

noLocal (entrée)
La valeur `True` signifie que le consommateur de message ne reçoit pas les messages publiés par sa propre connexion. La valeur `False` signifie que le consommateur de message reçoit les messages publiés par sa propre connexion. La valeur par défaut est `False`.

Retour :
Objet MessageConsumer.

Exceptions :

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateDurableSubscriber - Création d'un abonné durable

Interface :

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription) ;
```

Crée un abonné durable pour la rubrique spécifiée.

Cette méthode n'est pas valide pour une connexion en temps réel à un courtier.

Pour plus d'informations sur les abonnés durables, voir «Abonnés durables», à la page 34.

Paramètres :

dest (entrée)
Un objet Destination représentant la rubrique. Il ne doit pas s'agir d'une rubrique temporaire.

subscription (entrée)
Un objet String encapsulant un nom qui identifie l'abonnement durable. Le nom doit être unique dans l'identificateur de client pour la connexion.

Retour :
Objet MessageConsumer représentant l'abonné durable.

Exceptions :

- XMSException
- InvalidDestinationException

CreateDurableSubscriber - Création d'un abonné durable (avec sélecteur de message et indicateur de message local)

Interface :

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription,
                                         String selector,
                                         Boolean noLocal) ;
```

Crée un abonné durable pour la rubrique spécifiée avec un sélecteur de message et en indiquant si l'abonné durable reçoit les messages publiés par sa propre connexion.

Cette méthode n'est pas valide pour une connexion en temps réel à un courtier.

Pour plus d'informations sur les abonnés durables, voir «Abonnés durables», à la page 34.

Paramètres :

dest (entrée)
Un objet Destination représentant la rubrique. Il ne doit pas s'agir d'une rubrique temporaire.

subscription (entrée)
Un objet String encapsulant un nom qui identifie l'abonnement durable. Le nom doit être unique dans l'identificateur de client pour la connexion.

selector (entrée)
Objet String encapsulant une expression de sélecteur de message. Seuls les messages dont les propriétés correspondent à l'expression du sélecteur de message sont distribués sur l'abonné durable.

Un objet String Null signifie qu'aucun sélecteur de message n'est défini pour l'abonné durable.

noLocal (entrée)
La valeur `True` signifie que l'abonné durable ne reçoit pas les messages publiés par sa propre connexion. La valeur `False` signifie que l'abonné durable reçoit les messages publiés par sa propre connexion. La valeur par défaut est `False`.

Retour :
Objet MessageConsumer représentant l'abonné durable.

Exceptions :

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateMapMessage - Création d'un message de mappe

Interface :

```
IMapMessage CreateMapMessage();
```

Crée un message de mappe.

Paramètres :
Aucun

Retour :
Objet MapMessage.

Exceptions :

- XMSException
- IllegalStateException (The session is closed)

CreateMessage - Création d'un message

Interface :

```
IMessage CreateMessage();
```

Crée un message sans corps.

Paramètres :
Aucun

Retour :
Objet Message.

Exceptions :

- XMSException
- IllegalStateException (The session is closed)

CreateObjectMessage - Création d'un message d'objet

Interface :

```
IObjectMessage CreateObjectMessage();
```

Crée un message d'objet.

Paramètres :
Aucun

Retour :
Objet ObjectMessage.

Exceptions :

- XMSException
- IllegalStateException (The session is closed)

CreateProducer - Création d'un expéditeur

Interface :

```
IMessageProducer CreateProducer(IDestination dest) ;
```

Crée un expéditeur de message pour envoyer des messages vers la destination spécifiée.

Paramètres :

dest (entrée)
Objet Destination.

Si vous spécifiez un objet Destination Null, l'expéditeur de message est créé sans destination. Dans ce cas, l'application doit spécifier une destination chaque fois qu'elle utilise l'expéditeur de message pour envoyer un message.

Retour :
Objet MessageProducer.

Exceptions :

- XMSException
- InvalidDestinationException

CreateQueue - Création d'une file d'attente

Interface :

```
IDestination CreateQueue(String queue) ;
```

Crée un objet Destination pour représenter la file d'attente sur le serveur de messagerie.

Cette méthode ne permet pas de créer la file d'attente sur le serveur de messagerie. Vous devez créer celle-ci avant qu'une application puisse appeler cette méthode.

Paramètres :

queue (entrée)
Un objet de type String encapsulant le nom de la file d'attente ou un identificateur URI (Uniform Resource Identifier) qui identifie la file d'attente.

Retour :
Objet Destination représentant la file d'attente.

Exceptions :

- XMSException

CreateStreamMessage - Création d'un message de flux

Interface :

```
IStreamMessage CreateStreamMessage();
```

Crée un message de flux.

Paramètres :
Aucun

Retour :
Objet StreamMessage.

Exceptions :

- XMSException
- XMS_ILLEGAL_STATE_EXCEPTION

CreateTemporaryQueue - Création d'une file d'attente temporaire

Interface :

```
IDestination CreateTemporaryQueue() ;
```

Crée une file d'attente temporaire.

La portée de la file d'attente temporaire est la connexion. Seules les sessions créées par la connexion peuvent utiliser la file d'attente temporaire.

La file d'attente temporaire reste active jusqu'à sa suppression explicite ou la fin de la connexion.

Pour plus d'informations sur les files d'attente temporaires, voir «Destinations temporaires», à la page 32.

Paramètres :
Aucun

Retour :
Objet Destination représentant la file d'attente temporaire.

Exceptions :

- XMSException

CreateTemporaryTopic - Création d'une rubrique temporaire

Interface :

```
IDestination CreateTemporaryTopic() ;
```

Crée une rubrique temporaire.

La portée de la rubrique temporaire est la connexion. Seules les sessions créées par la connexion peuvent utiliser la rubrique temporaire.

La rubrique temporaire reste active jusqu'à sa suppression explicite ou la fin de la connexion.

Pour plus d'informations sur les rubriques temporaires, voir «Destinations temporaires», à la page 32.

Paramètres :
Aucun

Retour :
Objet Destination représentant la rubrique temporaire.

Exceptions :

- XMSException

CreateTextMessage - Création d'un message texte

Interface :

```
ITextMessage CreateTextMessage();
```

Crée un message texte avec un corps vide.

Paramètres :
Aucun

Retour :
Objet TextMessage.

Exceptions :

- XMSException

CreateTextMessage - Création d'un message texte (initialisé)

Interface :

```
ITextMessage CreateTextMessage(String initialValue);
```

Crée un message texte dont le corps est initialisé avec le texte spécifié.

Paramètres :

initialValue (entrée)
Objet String encapsulant le texte qui initialise le corps du message texte.

Aucun

Retour :
Objet TextMessage.

Exceptions :

- XMSException

CreateTopic - Création d'une rubrique

Interface :

```
IDestination CreateTopic(String topic) ;
```

Crée un objet Destination pour représenter une rubrique.

Paramètres :

topic (entrée)
Objet String encapsulant le nom de la rubrique ou un identificateur URI qui identifie la rubrique.

Retour :
Objet Destination représentant la rubrique.

Exceptions :

- XMSException

Recover - Restauration

Interface :

```
void Recover();
```

Restaure la session. La livraison du message est arrêtée puis reprise avec le message le plus ancien sans accusé de réception.

La session ne doit pas être transactionnelle.

Pour plus d'informations sur la restauration d'une session, voir «Accusé de réception de message», à la page 26.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException
- IllegalStateException

Rollback - Annulation

Interface :

```
void Rollback();
```

Annule tous les messages traités lors de la transaction en cours.

La session doit être transactionnelle.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException
- IllegalStateException

Unsubscribe - Désabonnement

Interface :

```
void Unsubscribe(String subscription);
```

Supprime un abonnement durable. Le serveur de messagerie supprime l'enregistrement de l'abonnement durable qu'il gère et n'envoie plus de messages à l'abonné durable.

Une application ne peut pas supprimer un abonnement durable dans les circonstances suivantes :

- Un consommateur de message est actif pour l'abonnement durable
- Un message consommé fait partie d'une transaction en attente
- Aucun accusé de réception n'a été émis pour un message consommé

Cette méthode n'est pas valide pour une connexion en temps réel à un courtier.

Paramètres :

subscription (entrée)
Objet String encapsulant le nom qui identifie l'abonnement durable.

Retour :
Vide

Exceptions :

- XMSException
- InvalidDestinationException
- IllegalStateException

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IStreamMessage

Un message de flux est un message dont le corps contient un flux de valeurs et où chaque valeur a un type de données associé. Le contenu du corps est écrit et lu séquentiellement.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IStreamMessage
```

Lorsqu'une application lit une valeur à partir du flux de message, celle-ci est convertie par XMS dans un autre type de données. Pour plus d'informations sur cette forme de conversion implicite, voir «Messages de flux», à la page 80.

Référence associée

Messages de flux

Le corps d'un message de flux contient un flux de valeurs, chacune d'entre elles ayant un type de données associé.

Des méthodes

ReadBoolean - Lecture de valeur booléenne

Interface :

```
Boolean ReadBoolean();
```

Lit une valeur booléenne depuis le flux de message.

Paramètres :
Aucun

Retour :
Valeur booléenne lue.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Lecture d'un octet

Interface :

```
Int16   ReadSignedByte();
    Byte    ReadByte();
```

Lit un entier signé de 8 bits depuis le flux de message.

Paramètres :
Aucun

Retour :
Octet lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Lecture d'octets

Interface :

```
Int32   ReadBytes(Byte[] array);
```

Lit un tableau d'octets depuis le flux de message.

Paramètres :

array (entrée)
Mémoire tampon contenant le tableau d'octets qui est lu et longueur de la mémoire tampon en octets.

Si le nombre d'octets du tableau est inférieur ou égal à la longueur de la mémoire tampon, l'ensemble du tableau est lu dans la mémoire tampon. Si le nombre d'octets du tableau est supérieur ou égal à la longueur de la mémoire tampon, celle-ci est remplie avec une partie du tableau et le curseur interne marque la position de l'octet suivant à lire. Un appel ultérieur à readBytes() lit les octets du tableau à partir de la position actuelle du curseur.

Si vous spécifiez un pointeur Null en entrée, l'appel ignore le tableau et ne le lit pas.

Retour :
Nombre d'octets lus dans la mémoire tampon. Si la mémoire tampon est partiellement remplie, la valeur est inférieure à la longueur de la mémoire tampon, indiquant qu'il ne reste plus d'octets à lire dans le tableau. S'il ne reste plus d'octets à lire à partir du tableau avant l'appel, la valeur est XMSC_END_OF_BYTEARRAY.

Si vous spécifiez un pointeur Null en entrée, la méthode ne renvoie aucune valeur.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadChar - Lecture de caractère

Interface :

```
Char    ReadChar();
```

Lit un caractère de 2 octets depuis le flux de message.

Paramètres :
Aucun

Retour :
Caractère lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Lecture d'un nombre en virgule flottante à double précision

Interface :

```
Double  ReadDouble();
```

Lit un nombre en virgule flottante à double précision à partir du flux de message.

Paramètres :
Aucun

Retour :
Le nombre en virgule flottante à double précision lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Lecture du nombre en virgule flottante

Interface :

```
Single  ReadFloat();
```

Lit un nombre en virgule flottante à 4 octets à partir du flux de message.

Paramètres :
Aucun

Retour :
Le nombre en virgule flottante lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Lecture d'un entier

Interface :

```
Int32   ReadInt();
```

Lit un entier signé de 32 bits depuis le flux de message.

Paramètres :
Aucun

Retour :
Entier lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Lecture d'un entier long

Interface :

```
Int64   ReadLong();
```

Lit un entier signé de 64 bits depuis le flux de message.

Paramètres :
Aucun

Retour :
Entier long lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadObject - Lecture d'objet

Interface :

```
Object  ReadObject();
```

Lit une valeur à partir du flux de message et renvoie son type de données.

Paramètres :
Aucun

Retour :
La valeur, dont le type d'objet peut être :

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Exceptions :
XMSException

ReadShort - Lecture d'un entier court

Interface :

```
Int16   ReadShort();
```

Lit un entier signé de 16 bits depuis le flux de message.

Paramètres :
Aucun

Retour :
Entier court lu.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadString - Lecture de chaîne

Interface :

```
String  ReadString();
```

Lit un chaîne depuis le flux de messages. Si nécessaire, XMS convertit les caractères en chaîne dans la page de codes locale.

Paramètres :
Aucun

Retour :
Objet String encapsulant la chaîne lue. Si une conversion de données est requise, il s'agit de la chaîne après conversion.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - Réinitialisation

Interface :

```
void  Reset();
```

Met le corps du message en mode lecture seule et repositionne le curseur au début du flux de message.

Paramètres :
Aucun

Retour :
Vide

Exceptions :

- XMSException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - Ecriture d'une valeur booléenne

Interface :

```
void  WriteBoolean(Boolean value);
```

Ecrit une valeur booléenne sur un flux de message.

Paramètres :

value (entrée)
Valeur booléenne à écrire.

Retour :
Vide

Exceptions :

- XMSException

- MessageNotWritableException

WriteByte - Ecriture d'un octet

Interface :

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Ecrit un octet sur le flux de message.

Paramètres :

value (entrée)
Octet à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteBytes - Ecriture d'octets

Interface :

```
void  WriteBytes(Byte[] value);
```

Ecrit un tableau d'octets sur un flux de message.

Paramètres :

value (entrée)
Tableau des octets à écrire.

length (entrée)
Nombre d'octets dans le tableau.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteChar - Ecriture d'un caractère

Interface :

```
void  WriteChar(Char value);
```

Ecrit un caractère sur le flux de message sous forme de 2 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Caractère à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteDouble - Ecriture d'un nombre en virgule flottante à double précision

Interface :

```
void  WriteDouble(Double value);
```

Convertit un nombre en virgule flottante à double précision en entier long et écrit celui-ci sur le flux de message d'octets sous forme de 8 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Le nombre en virgule flottante à double précision à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteFloat - Ecriture d'un nombre en virgule flottante

Interface :

```
void  WriteFloat(Single value);
```

Convertit un nombre en virgule flottante en entier et écrit cet entier sur le flux de message sous forme de 4 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Le nombre en virgule flottante à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteInt - Ecriture d'un entier

Interface :

```
void  WriteInt(Int32 value);
```

Ecrit un entier sur le flux de message d'octets sous forme de 4 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Entier à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteLong - Ecriture d'un entier long

Interface :

```
void  WriteLong(Int64 value);
```

Ecrit un entier long sur le flux de message sous forme de 8 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Entier long à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteObject - Ecriture d'un objet

Interface :

```
void  WriteObject(Object value);
```

Ecrit une valeur, avec un type de données spécifié, sur le flux de message.

Paramètres :

objectType (entrée)
La valeur, dont le type d'objet peut être :

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value (entrée)
Tableau d'octets contenant la valeur à écrire.

length (entrée)
Nombre d'octets dans le tableau.

Retour :
Vide

Exceptions :

- XMSException

WriteShort - Ecriture d'un entier court

Interface :

```
void  WriteShort(Int16 value);
```

Ecrit un entier court sur le flux de message sous forme de 2 octets, l'octet de poids fort en premier.

Paramètres :

value (entrée)
Entier court à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

WriteString - Ecriture d'une chaîne

Interface :

```
void  WriteString(String value);
```

Ecrit une chaîne sur le flux de message.

Paramètres :

value (entrée)
Objet String encapsulant la chaîne à écrire.

Retour :
Vide

Exceptions :

- XMSException
- MessageNotWritableException

Propriétés et méthodes héritées

Les propriétés suivantes sont héritées de l'interface IMessage :

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Les méthodes suivantes sont héritées de l'interface IMessage :

clearBody, clearProperties, PropertyExists

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ITextMessage

Un message texte est un message dont le corps contient une chaîne.

Hiérarchie d'héritage :

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.ITextMessage
```

Référence associée

Messages texte
Le corps d'un message texte contient une chaîne.

.NET propriétés

Text - Extraction et définition de texte

Interface :

```
String Text
{
  get;
  set;
}
```

Extrait et défini la chaîne qui forme le corps du message texte.

Si nécessaire, XMS convertit les caractères en chaîne dans la page de codes locale.

Exceptions :

- XMSException
- MessageNotReadableException
- MessageNotWritableException
- MessageEOFException

Propriétés et méthodes héritées

Les propriétés suivantes sont héritées de l'interface IMessage :

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Les méthodes suivantes sont héritées de l'interface IMessage :

clearBody, clearProperties, PropertyExists

Les méthodes suivantes sont héritées de l'interface IPropertyContext :

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty,

SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

TransactionInProgressException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionInProgressException
```

XMS émet cette exception si une application demande une opération qui n'est pas valide car une transaction est en cours.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

TransactionRolledBackException

Hiérarchie d'héritage :

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionRolledBackException
```

XMS émet cette exception si une application appelle Session.commit() pour valider la transaction en cours, mais que la transaction est ensuite annulée.

Propriétés et méthodes héritées

Les méthodes suivantes sont héritées de l'interface XMSException :

GetErrorCode, GetLinkedException

XMSException

Si XMS détecte une erreur lors du traitement d'un appel à une méthode .NET , XMS émet une exception. Une exception est un objet qui encapsule des informations relatives à l'erreur.

Hiérarchie d'héritage :

```
System.Exception
   |
   +----IBM.XMS.XMSException
```

Il existe différents types d'exception XMS et un objet XMSException n'est qu'un seul type d'exception. Toutefois, la classe XMSException est une superclasse des classes d'exception XMS. XMS émet un objet XMSException si aucun autre type d'exception n'est approprié.

.NET propriétés

ErrorCode - Extraction du code d'erreur

Interface :

```
public String ErrorCode
{
  get {return errorCode_;}
}
```

Extrait le code d'erreur.

Exceptions :

- XMSException

LinkedException - Extraction de l'exception associée

Interface :

```
public Exception LinkedException
{
  get { return linkedException_;}
  set { linkedException_ = value;}
}
```

Extrait l'exception suivante dans la chaîne d'exceptions.

La méthode renvoie une valeur Null s'il n'y a plus d'exception dans la chaîne.

Exceptions :

- XMSException

XMSFactoryFactory

Si une application n'utilise pas les objets gérés, utilisez cette classe pour créer des fabriques de connexions, des files d'attente et des rubriques.

Hiérarchie d'héritage :
Aucun

.NET propriétés

Metadata - Extraction de métadonnées

Interface :

```
IConnectionMetaData MetaData
```

Extrait les métadonnées appropriées pour le type de connexion de l'objet XMSFactoryFactory.

Exceptions :
Aucun

Des méthodes

CreateConnectionFactory - Création d'une fabrique de connexions

Interface :

```
IConnectionFactory CreateConnectionFactory();
```

Crée un objet ConnectionFactory du type déclaré.

Paramètres :
Aucun

Retour :
Objet ConnectionFactory.

Exceptions :

- XMSException

CreateQueue - Création d'une file d'attente

Interface :

```
IDestination CreateQueue(String name);
```

Crée un objet Destination pour représenter la file d'attente sur le serveur de messagerie.

Cette méthode ne permet pas de créer la file d'attente sur le serveur de messagerie. Vous devez créer celle-ci avant qu'une application puisse appeler cette méthode.

Paramètres :

name (entrée)
Un objet de type String encapsulant le nom de la file d'attente ou un identificateur URI (Uniform Resource Identifier) qui identifie la file d'attente.

Retour :
Objet Destination représentant la file d'attente.

Exceptions :

- XMSException

CreateTopic - Création d'une rubrique

Interface :

```
IDestination CreateTopic(String name);
```

Crée un objet Destination pour représenter une rubrique.

Paramètres :

name (entrée)
Objet String encapsulant le nom de la rubrique ou un identificateur URI qui identifie la rubrique.

Retour :
Objet Destination représentant la rubrique.

Exceptions :

- XMSException

GetInstance - Extraction d'une instance de XMSFactoryFactory

Interface :

```
static XMSFactoryFactory GetInstance(int connectionType);
```

Créez une instance de XMSFactoryFactory. Une application XMS utilise un objet XMSFactoryFactory pour obtenir une référence à un objet ConnectionFactory approprié pour le type de protocole requis. Cet objet ConnectionFactory peut ensuite produire des connexions uniquement pour ce type de protocole.

Paramètres :

connectionType (entrée)
Le type de connexion pour laquelle l'objet ConnectionFactory génère des connexions :

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

Retour :
Objet XMSFactoryFactory dédié au type de connexion déclaré.

Exceptions :

- NotSupportedException

Propriétés des objets XMS

Cette chapitre présente les propriétés d'objet définies par XMS.

Cette chapitre contient les sections suivantes :

Chaque section répertorie les propriétés d'un objet du type spécifié et fournit une brève description de chaque propriété.

Ce fichier chapitre contient également le fichier «Définitions de propriété», à la page 198 section, qui fournit une définition de chaque propriété.

Si une application définit ses propres propriétés pour les objets décrits dans ce chapitre, cela ne génère pas d'erreur mais peut entraîner des résultats imprévisibles.

Remarque : Les noms de propriété et les valeurs de cette section sont affichés sous la forme XMSC.*NAME*, qui est la forme utilisée pour C et C + +. Toutefois, dans .NET, la forme du nom de propriété peut être XMSC.*NAME* ou XMSC_*NAME*, selon la façon dont vous l'utilisez:

- Si vous spécifiez une propriété, le nom de la propriété doit être au format XMSC.*NAME* , comme illustré dans l'exemple suivant:

```
cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
```

- Si vous spécifiez une chaîne, le nom de la propriété doit être au format XMSC_*NAME* , comme illustré dans l'exemple suivant:

```
cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
```

  Dans .NET, les noms et les valeurs de propriété sont fournis en tant que constantes dans la classe XMSC. Ces constantes identifient des chaînes et sont utilisées par toute application XMS .NET . Si vous utilisez ces constantes prédéfinies, les noms et les valeurs de propriété sont au format XMSC.*NOM* ; ainsi, vous utiliserez XMSC.USERID plutôt que XMSC_USERID.

Les types de données sont également au format utilisé pour C/C + +. Vous trouverez les valeurs correspondantes pour .NET dans«Types de données pour .NET», à la page 45.

Concepts associés

Génération de vos propres applications
Vous pouvez générer vos propres applications comme vous générez les modèles d'application.

Référence associée

.NET interfaces
Cette section documente les interfaces de classe .NET , ainsi que leurs propriétés et leurs méthodes.

Propriétés de Connection

Présentation des propriétés de l'objet Connection, avec des liens vers des informations de référence plus détaillées.

Tableau 25. Propriétés de Connection

Nom de la propriété	Description
«XMSC_WMQ_RESOLVED_QUEUE_MANAGER», à la page 233	Cette propriété est utilisée pour obtenir le nom du gestionnaire de files d'attente auquel il est connecté.
«XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID », à la page 233	Cette propriété est remplie avec l'ID du gestionnaire de files d'attente après la connexion.
XMSC_WPM_CONNECTION_PROTOCOL	Protocole de communication utilisé pour la connexion au moteur de messagerie. Cette propriété est en lecture seule.
XMSC_WPM_HOST_NAME	Nom hôte ou adresse IP du système qui contient le moteur de messagerie auquel l'application est connectée. Cette propriété est en lecture seule.
XMSC_WPM_ME_NAME	Nom du moteur de messagerie auquel l'application est connectée. Cette propriété est en lecture seule.
XMSC_WPM_PORT	Numéro du port écouté par le moteur de messagerie auquel l'application est connectée. Cette propriété est en lecture seule.

Un objet Connection a également des propriétés en lecture seule dérivées des propriétés de la fabrique de connexions qui a été utilisée pour créer la connexion. Ces propriétés sont dérivées non seulement des propriétés de fabrique de connexions définies lors de la création de la connexion, mais aussi des valeurs par défaut des propriétés qui n'ont pas été définies. Les propriétés incluent uniquement celles qui sont pertinentes pour le type de serveur de messagerie auquel l'application est connectée. Les noms de propriété sont les mêmes que les noms des propriétés de fabrique de connexions.

Propriétés de ConnectionFactory

Présentation des propriétés de l'objet ConnectionFactory, avec des liens vers des informations de référence plus détaillées.

Tableau 26. Propriétés de ConnectionFactory	
Nom de la propriété	**Description**
«XMSC_ASYNC_EXCEPTIONS», à la page 208	Cette propriété détermine si XMS informe un programme d'écoute des exceptions uniquement lorsqu'une connexion est interrompue, ou également lorsqu'une exception est émise de façon asynchrone sur un appel d'API XMS. Cette propriété s'applique à toutes les connexions créées à partir de la fabrique de connexions et pour lesquelles un programme d'écoute des exceptions est enregistré.
XMSC_CLIENT_ID	Identificateur de client pour une connexion.
XMSC_CONNECTION_TYPE	Type de serveur de messagerie auquel une application se connecte.
XMSC_PASSWORD	Mot de passe pouvant être utilisé pour authentifier l'application lorsqu'elle tente de se connecter à un serveur de messagerie.
«XMSC_RTT_BROKER_PING_INTERVAL», à la page 214	Intervalle de temps, en millisecondes, après lequel XMS .NET vérifie la connexion à un serveur de messagerie en temps réel pour détecter une activité.
XMSC_RTT_CONNECTION_PROTOCOL	Protocole de communication utilisé pour une connexion en temps réel à un courtier.
XMSC_RTT_HOST_NAME	Nom d'hôte ou adresse IP du système sur lequel un courtier s'exécute.
XMSC_RTT_LOCAL_ADDRESS	Nom d'hôte ou adresse IP de l'interface de réseau local à utiliser pour une connexion en temps réel à un courtier.
XMSC_RTT_MULTICAST	Paramètre de multidiffusion pour une fabrique de connexions ou une destination.
XMSC_RTT_PORT	Numéro du port sur lequel un courtier écoute les demandes entrantes.
XMSC_USERID	ID utilisateur pouvant être utilisé pour authentifier l'application lorsqu'elle tente de se connecter à un serveur de messagerie.
XMSC_WMQ_BROKER_CONTROLQ	Nom de la file d'attente de contrôle utilisée par un courtier. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.

Tableau 26. Propriétés de ConnectionFactory (suite)

Nom de la propriété	Description
XMSC_WMQ_BROKER_PUBQ	Nom de la file d'attente contrôlée par un courtier lorsque des applications envoient les messages qu'elles publient. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
XMSC_WMQ_BROKER_QMGR	Nom du gestionnaire de files d'attente auquel un courtier est connecté. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
XMSC_WMQ_BROKER_SUBQ	Nom de la file d'attente de souscription pour un consommateur de message non durable. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
XMSC_WMQ_BROKER_VERSION	Type de courtier utilisé par l'application pour une connexion ou pour la destination. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
«XMSC_WMQ_CCDTURL», à la page 219	Adresse URL qui identifie le nom et l'emplacement du fichier contenant la table de définition de canal du client et qui spécifie le mode d'accès au fichier.
XMSC_WMQ_CHANNEL	Nom du canal à utiliser pour une connexion.
«XMSC_WMQ_CLIENT_RECONNECT_OPTIONS», à la page 219	Cette propriété spécifie les options de reconnexion client pour les nouvelles connexions créées par cette fabrique
«XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT», à la page 220	Cette propriété spécifie le temps, en secondes, qu'un connexion client utilise pour essayer de se reconnecter.
XMSC_WMQ_CONNECTION_MODE	Mode par lequel une application se connecte à un gestionnaire de files d'attente.

Tableau 26. Propriétés de ConnectionFactory (suite)	
Nom de la propriété	**Description**
«XMSC_WMQ_CONNECTION_NAME_LIST», à la page 221	Cette propriété spécifie les hôtes auxquels le client tente de se reconnecter après une interruption de sa connexion.
XMSC_WMQ_FAIL_IF_QUIESCE	Indique que les appels à certaines méthodes échouent si le gestionnaire de files d'attente auquel l'application est connectée est à l'état de mise au repos.
XMSC_WMQ_HOST_NAME	Nom d'hôte ou adresse IP du système sur lequel s'exécute un gestionnaire de files d'attente.
XMSC_WMQ_LOCAL_ADDRESS	Pour une connexion à un gestionnaire de files d'attente, cette propriété spécifie l'interface de réseau local et/ou le port ou la plage de ports à utiliser.
XMSC_WMQ_MESSAGE_SELECTION	Détermine si la sélection des messages est effectuée par le client XMS ou par le courtier. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
XMSC_WMQ_MSG_BATCH_SIZE	Nombre maximal de messages à extraire d'une file d'attente en un seul lot lorsque la distribution asynchrone de messages est utilisée. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
XMSC_WMQ_POLLING_INTERVAL	Si chaque programme d'écoute de message dans une session ne comporte pas de message approprié dans sa file d'attente, cette valeur est l'intervalle de temps maximal, en millisecondes, qui s'écoule avant que chaque programme d'écoute de message essaie à nouveau d'obtenir un message à partir de sa file d'attente. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
«XMSC_WMQ_PROVIDER_VERSION», à la page 229	Version, édition, niveau de modification et groupe de correctifs du gestionnaire de files d'attente auquel l'application a l'intention de se connecter.
XMSC_WMQ_PORT	Numéro du port sur lequel un gestionnaire de files d'attente écoute les demandes entrantes.

Tableau 26. Propriétés de ConnectionFactory (suite)

Nom de la propriété	Description
XMSC_WMQ_PUB_ACK_INTERVAL	Nombre de messages publiés par un diffuseur de publications avant que le client XMS ne demande un accusé de réception du courtier. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
«XMSC_WMQ_PUT_ASYNC_ALLOWED», à la page 225	Cette propriété détermine si les expéditeurs de message sont autorisés à utiliser les insertions asynchrones pour envoyer des messages à cette destination.
XMSC_WMQ_QMGR_CCSID	Identificateur (CCSID) du jeu de caractères codés, ou page de codes, dans lequel les zones de données de type caractères définies dans l'interface MQI (Message Queue Interface) sont échangées entre le client XMS et le client WebSphere MQ .
XMSC_WMQ_QUEUE_MANAGER	Nom du gestionnaire de files d'attente auquel se connecter.
XMSC_WMQ_RECEIVE_EXIT	Identifie un exit de réception de canal à exécuter.
XMSC_WMQ_RECEIVE_EXIT_INIT	Données utilisateur transmises à un exit de réception de canal lors de l'appel.
XMSC_WMQ_SECURITY_EXIT	Identifie un exit de sécurité de canal.
XMSC_WMQ_SECURITY_EXIT_INIT	Données utilisateur transmises à un exit de sécurité de canal lors de l'appel.
«XMSC_WMQ_SEND_CHECK_COUNT», à la page 234	Nombre d'appels d'envoi à autoriser entre deux vérifications d'erreurs asynchrones, au cours d'une même session XMS non transactionnelle.
XMSC_WMQ_SEND_EXIT	Identifie un exit d'émission de canal.
XMSC_WMQ_SEND_EXIT_INIT	Données utilisateur qui sont transmises aux exits d'émission une fois ceux-ci appelés.
«XMSC_WMQ_SHARE_CONV_ALLOWED», à la page 234	Indique si une connexion client peut partager son socket avec d'autres connexions XMS de niveau supérieur à partir du même processus vers le même gestionnaire de files d'attente, si les définitions de canal correspondent. Cette propriété est fournie pour permettre un isolement complet des connexions dans des sockets distincts, lorsque cela est requis pour des raisons de développement, de maintenance ou de fonctionnement.
XMSC_WMQ_SSL_CERT_STORES	Emplacements des serveurs contenant les listes de révocation de certificat à utiliser sur une connexion SSL à un gestionnaire de files d'attente.
XMSC_WMQ_SSL_CIPHER_SPEC	Nom de la spécification CipherSpec à utiliser sur une connexion sécurisée vers un gestionnaire de files d'attente.

Tableau 26. Propriétés de ConnectionFactory (suite)	
Nom de la propriété	**Description**
XMSC_WMQ_SSL_CIPHER_SUITE	Nom de la suite de chiffrement à utiliser sur une connexion TLS à un gestionnaire de files d'attente. Le protocole utilisé lors de la négociation de la connexion sécurisée dépend de la suite de chiffrement spécifiée.
XMSC_WMQ_SSL_CRYPTO_HW	Détails de configuration pour le matériel de cryptographie connecté au système client.
XMSC_WMQ_SSL_FIPS_REQUIRED	La valeur de cette propriété détermine si une application peut ou non utiliser des suites de chiffrement conformes non FIPS. Si cette propriété est pour valeur true, seuls les algorithmes FIPS sont utilisés pour la connexion client/serveur.
XMSC_WMQ_SSL_KEY_REPOSITORY	Emplacement du fichier de clés dans lequel les clés et les certificats sont stockés.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Nombre total d'octets non chiffrés envoyés et reçus dans une conversation SSL avant renégociation de la clé confidentielle.
XMSC_WMQ_SSL_PEER_NAME	Nom d'homologue à utiliser sur une connexion SSL vers un gestionnaire de files d'attente.
XMSC_WMQ_SYNCPOINT_ALL_GETS	Indique si tous les messages doivent être extraits des files d'attente sous le contrôle d'un point de synchronisation.
«XMSC_WMQ_TARGET_CLIENT», à la page 241	
XMSC_WMQ_TEMP_Q_PREFIX	Préfixe utilisé pour former le nom de la file d'attente dynamique WebSphere MQ créée lorsque l'application crée une file d'attente temporaire Un XMS .
XMSC_WMQ_TEMP_TOPIC_PREFIX	Lors de la création de rubriques temporaires, XMS génère une chaîne de rubrique au format "TEMP/TEMPTOPICPREFIX/unique_id" ou, si cette propriété contient la valeur par défaut, cette chaîne, "TEMP/unique_id", est générée. La spécification d'une valeur non vide permet la définition de files d'attente modèle spécifiques pour la création des files d'attente gérées pour les abonnés à des rubriques temporaires créées sous cette connexion.
XMSC_WMQ_TEMPORARY_MODEL	Nom de la file d'attente modèle WebSphere MQ à partir de laquelle une file d'attente dynamique est créée lorsque l'application crée une file d'attente temporaire Un XMS .
XMSC_WPM_BUS_NAME	Pour une fabrique de connexions, nom du bus d'intégration de services auquel l'application se connecte ; pour une destination, nom du bus d'intégration de services dans lequel la destination existe.
XMSC_WPM_CONNECTION_PROXIMITY	Paramètre de proximité de connexion pour la connexion.
XMSC_WPM_DUR_SUB_HOME	Nom du moteur de messagerie où sont gérés les abonnements durables pour une connexion ou une destination.

Tableau 26. Propriétés de ConnectionFactory (suite)	
Nom de la propriété	**Description**
XMSC_WPM_LOCAL_ADDRESS	Pour une connexion à un bus d'intégration de services, cette propriété spécifie l'interface de réseau local et/ou le port ou la plage de ports à utiliser.
XMSC_WPM_NON_PERSISTENT_MAP	Niveau de fiabilité des messages non persistants envoyés à l'aide de la connexion.
XMSC_WPM_PERSISTENT_MAP	Niveau de fiabilité des messages non persistants envoyés à l'aide de la connexion.
XMSC_WPM_PROVIDER_ENDPOINTS	Séquence d'une ou plusieurs adresses de noeud final de serveurs d'amorçage.
XMSC_WPM_TARGET_GROUP	Nom d'un groupe cible de moteurs de messagerie.
XMSC_WPM_TARGET_SIGNIFICANCE	Signification du groupe cible de moteurs de messagerie.
XMSC_WPM_TARGET_TRANSPORT_CHAIN	Nom de la chaîne de transport entrant que l'application doit utiliser pour se connecter à un moteur de messagerie.
XMSC_WPM_TARGET_TYPE	Type du groupe cible de moteurs de messagerie.
XMSC_WPM_TEMP_Q_PREFIX	Préfixe utilisé pour former le nom de la file d'attente temporaire créée dans le bus d'intégration de services lorsque l'application crée une file d'attente temporaire Un XMS .
XMSC_WPM_TEMP_TOPIC_PREFIX	Préfixe utilisé pour former le nom d'une rubrique temporaire créée par l'application.

Concepts associés

Objets ConnectionFactories et Connection
Un objet ConnectionFactory fournit un modèle utilisé par une application pour créer un objet Connection. L'application utilise l'objet Connection pour créer un objet Session.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Connexions sécurisées à un gestionnaire de files d'attente IBM MQ
Pour permettre à une application XMS .NET de créer des connexions sécurisées à un gestionnaire de files d'attente IBM MQ, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Connexions sécurisées à un moteur de messagerie WebSphere Application Server service integration bus
Pour permettre à une application XMS .NET de créer des connexions sécurisées à un moteur de messagerie de WebSphere Application Server service integration bus, les propriétés pertinentes doivent être définies dans l'objet ConnectionFactory.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

Propriétés requises pour les objets ConnectionFactory gérés
Lorsqu'une application crée une fabrique de connexions, un certain nombre de propriétés doivent être définies pour la création d'une connexion à un serveur de messagerie.

Propriétés de ConnectionMetaData

Présentation des propriétés de l'objet ConnectionMetaData, avec des liens vers des informations de référence plus détaillées.

Tableau 27. Propriétés de ConnectionMetaData

Nom de la propriété	Description
XMSC_JMS_MAJOR_VERSION	Numéro de version principale de la spécification JMS sur laquelle repose XMS . Cette propriété est en lecture seule.
XMSC_JMS_MINOR_VERSION	Numéro de version secondaire de la spécification JMS sur laquelle repose XMS . Cette propriété est en lecture seule.
XMSC_JMS_VERSION	Identificateur de version de la spécification JMS sur laquelle est basée XMS . Cette propriété est en lecture seule.
XMSC_MAJOR_VERSION	Numéro de version du client XMS . Cette propriété est en lecture seule.
XMSC_MINOR_VERSION	Numéro d'édition du client XMS . Cette propriété est en lecture seule.
XMSC_PROVIDER_NAME	Fournisseur du client XMS . Cette propriété est en lecture seule.
XMSC_VERSION	Identificateur de version du client XMS . Cette propriété est en lecture seule.

Propriétés de Destination

Présentation des propriétés de l'objet Destination, avec des liens vers des informations de référence plus détaillées.

Tableau 28. Propriétés de Destination

Nom de la propriété	Description
XMSC_DELIVERY_MODE	Mode de livraison des messages envoyés à la destination.
XMSC_PRIORITY	Priorité des messages envoyés à la destination.
XMSC_RTT_MULTICAST	Paramètre de multidiffusion pour une fabrique de connexions ou une destination.
XMSC_TIME_TO_LIVE	Durée de vie des messages envoyés à la destination.
XMSC_WMQ_BROKER_VERSION	Type de courtier utilisé par l'application pour une connexion ou pour la destination.
XMSC_WMQ_CCSID	Identificateur (CCSID) du jeu de caractères codés, ou page de codes, dans lequel se trouvent les chaînes de données de type caractères dans le corps d'un message lorsque le client XMS transmet le message à la destination.

Tableau 28. Propriétés de Destination (suite)

Nom de la propriété	Description
XMSC_WMQ_DUR_SUBQ	Nom de la file d'attente de souscription pour un abonné durable qui reçoit des messages de la destination. **Remarque :** Cette propriété peut être utilisée avec la version 2.0 de IBM Message Service Client for .NET mais n'a aucun effet sur une application connectée à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 , sauf si la propriété XMSC_WMQ_PROVIDER_VERSION de la fabrique de connexions est définie sur un numéro de version inférieur à 7.
XMSC_WMQ_ENCODING	Mode de représentation des données numériques dans le corps d'un message lorsque le client XMS transmet le message à la destination.
XMSC_WMQ_FAIL_IF_QUIESCE	Indique que les appels à certaines méthodes échouent si le gestionnaire de files d'attente auquel l'application est connectée est à l'état de mise au repos.
«XMSC_WMQ_MESSAGE_BODY», à la page 223	Cette propriété détermine si une application XMS traite l' MQRFH2 d'un message IBM WebSphere MQ dans la charge du message (c'est-à-dire dans le corps du message).
«XMSC_WMQ_MQMD_MESSAGE_CONTEXT», à la page 224	Détermine le niveau de contexte de message à définir par l'application XMS . L'application doit être exécutée avec les droits appropriés pour que cette propriété soit appliquée.
«XMSC_WMQ_MQMD_READ_ENABLED», à la page 224	Cette propriété détermine si une application XMS peut ou non extraire les valeurs des zones MQMD.
«XMSC_WMQ_MQMD_WRITE_ENABLED», à la page 225	Cette propriété détermine si une application XMS peut extraire les valeurs des zones MQMD.
«XMSC_WMQ_READ_AHEAD_ALLOWED», à la page 226	Cette propriété détermine si les consommateurs de message et les navigateurs de files d'attente sont autorisés à utiliser la lecture anticipée pour extraire les messages non permanents et non transactionnels de cette destination dans une mémoire tampon interne avant de les recevoir.
«XMSC_WMQ_READ_AHEAD_CLOSE_POLICY», à la page 226	Cette propriété détermine, pour les messages distribués à un programme d'écoute de messages asynchrone, ce qu'il advient des messages dans la mémoire tampon interne de lecture anticipée lorsque le consommateur de messages est fermé.
«XMSC_WMQ_RECEIVE_CCSID», à la page 231	Propriété de destination qui définit le CCSID cible pour la conversion des messages du gestionnaire de files d'attente. La valeur est ignorée sauf si XMSC_WMQ_RECEIVE_CONVERSION est défini sur WMQ_RECEIVE_CONVERSION_QMGR.
«XMSC_WMQ_RECEIVE_CONVERSION», à la page 232	Propriété de destination qui détermine si la conversion de données sera effectuée par le gestionnaire de files d'attente.
XMSC_WMQ_TARGET_CLIENT	Indique si les messages envoyés à la destination contiennent un en-tête MQRFH2.

Tableau 28. Propriétés de Destination (suite)	
Nom de la propriété	**Description**
XMSC_WMQ_TEMP_TOPIC_PREFIX	Lors de la création de rubriques temporaires, XMS génère une chaîne de rubrique au format "TEMP/TEMPTOPICPREFIX/unique_id" ou, si cette propriété contient la valeur par défaut, cette chaîne, "TEMP/unique_id", est générée. La spécification d'une valeur non vide permet la définition de files d'attente modèle spécifiques pour la création des files d'attente gérées pour les abonnés à des rubriques temporaires créées sous cette connexion.
XMSC_WPM_BUS_NAME	Pour une fabrique de connexions, nom du bus d'intégration de services auquel l'application se connecte ; pour une destination, nom du bus d'intégration de services dans lequel la destination existe.
XMSC_WPM_TOPIC_SPACE	Nom de l'espace de sujet contenant la rubrique.

Concepts associés

Objets ConnectionFactories et Connection
Un objet ConnectionFactory fournit un modèle utilisé par une application pour créer un objet Connection. L'application utilise l'objet Connection pour créer un objet Session.

Connexion à un bus d'intégration de services
Une application XMS peut se connecter à un bus d'intégration de services WebSphere Application Server via une connexion TCP/IP directe ou via HTTP sur TCP/IP.

Destinations
Une application XMS utilise un objet Destination pour spécifier la destination des messages envoyés et la source des messages reçus.

Caractères génériques de destination
XMS prend en charge les caractères génériques de destination et assure leur transmission vers l'endroit où ils sont requis pour une correspondance. Il existe un schéma de caractères génériques différent pour chaque type de serveur pouvant être associé à XMS.

identificateur URI de rubrique
L'identificateur URI (Uniform Resource Identifier) de rubrique spécifie le nom de la rubrique ; il peut également en spécifier une ou plusieurs propriétés.

Identificateurs URI de file d'attente
L'identificateur URI d'une file d'attente indique le nom de cette file d'attente ; il peut également en spécifier une ou plusieurs propriétés.

Destinations temporaires
Les applications XMS peuvent créer et utiliser des destinations temporaires.

Mappage de propriété pour les objets gérés
Pour permettre aux applications d'utiliser les définitions d'objet de destination et de fabrique de connexions IBM MQ JMS et WebSphere Application Server , les propriétés extraites de ces définitions doivent être mappées aux propriétés XMS correspondantes qui peuvent être définies sur les fabriques de connexions et les destinations XMS .

Tâches associées

Création d'objets gérés
La définition des objets ConnectionFactory et Destination, dont les applications XMS ont besoin pour établir une connexion à un serveur de messagerie, doit être effectuée à l'aide des outils d'administration appropriés.

Référence associée

Propriétés requises pour les objets Destination gérés

Une application qui crée une destination doit définir plusieurs propriétés pour un objet Destination géré.

Propriétés de InitialContext

Présentation des propriétés de l'objet InitialContext, avec des liens vers des informations de référence plus détaillées.

Tableau 29. Propriétés de InitialContext

Nom de la propriété	Description
XMSC_IC_PROVIDER_URL	Utilisé pour localiser le répertoire de désignation JNDI de sorte que le service de dénomination COS n'ait pas besoin d'être sur le même serveur que le service Web.
XMSC_IC_SECURITY_AUTHENTICATION	Basé sur l'interface de contexte Java SECURITY_AUTHENTICATION. Cette propriété est applicable uniquement au contexte d'affectation de classe de service.
XMSC_IC_SECURITY_CREDENTIALS	Basé sur l'interface de contexte Java SECURITY_CREDENTIALS. Cette propriété est applicable uniquement au contexte d'affectation de classe de service.
XMSC_IC_SECURITY_PRINCIPAL	Basé sur l'interface de contexte Java SECURITY_PRINCIPAL. Cette propriété est applicable uniquement au contexte d'affectation de classe de service.
XMSC_IC_SECURITY_PROTOCOL	Basé sur l'interface de contexte Java SECURITY_PROTOCOL Cette propriété est applicable uniquement au contexte d'affectation de classe de service.
XMSC_IC_URL	Pour les contextes LDAP et FileSystem, adresse du référentiel contenant les objets gérés. Pour les contextes d'affectation de nom de classe de service, adresse du service Web qui recherche les objets dans le répertoire.

Concepts associés

Propriétés InitialContext
Les paramètres du constructeur InitialContext incluent l'emplacement du référentiel d'objets gérés, indiqué sous la forme d'un identificateur URI. Pour qu'une application établisse une connexion au référentiel, il peut être nécessaire de fournir des informations complémentaires à celles de l'identificateur URI.

Format d'identificateur URI pour contexte initial XMS
L'emplacement du référentiel d'objets gérés est fourni sous la forme d'un identificateur URI. Son format dépend du type de contexte.

Extraction d'objets gérés
XMS extrait un objet géré du référentiel à l'aide de l'adresse fournie lors de la création de l'objet InitialContext ou dans ses propriétés.

Tâches associées

Objets InitialContext
Une application doit créer un contexte initial afin de créer une connexion au référentiel d'objets gérés et en extraire les objets requis.

Propriétés de Message

Présentation des propriétés de l'objet Message, avec des liens vers des informations de référence plus détaillées.

Tableau 30. Propriétés de Message

Nom de la propriété	Description
JMS_IBM_CHARACTER_SET	Identificateur (CCSID) du jeu de caractères codés, ou page de codes, dans lequel se trouvent les chaînes de données de type caractères dans le corps du message lorsque le client XMS transmet le message à la destination prévue. Dans XMS, cette propriété a une valeur numérique mappée vers le CCSID. Toutefois, cette propriété est basée sur une propriété JMS qui contient une valeur de type chaîne et mappe vers le jeu de caractères Java qui représente ce CCSID numérique.
JMS_IBM_ENCODING	Mode de représentation des données numériques dans le corps du message lorsque le client XMS transmet le message à la destination prévue.
JMS_IBM_EXCEPTIONMESSAGE	Texte décrivant la raison pour laquelle le message a été envoyé à la destination d'exception. Cette propriété est en lecture seule.
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	Nom de la destination du message avant que celui-ci soit envoyé à la destination d'exception.
JMS_IBM_EXCEPTIONREASON	Code anomalie indiquant la raison pour laquelle le message a été envoyé à la destination d'exception.
JMS_IBM_EXCEPTIONTIMESTAMP	Heure à laquelle le message a été envoyé à la destination d'exception.
JMS_IBM_FEEDBACK	Code qui indique la nature d'un message de rapport.
JMS_IBM_FORMAT	Nature des données d'application dans le message.
JMS_IBM_LAST_MSG_IN_GROUP	Indique si le message est le dernier d'un groupe de messages.
JMS_IBM_MSGTYPE	Type du message.
JMS_IBM_PUTAPPLTYPE	Type de l'application ayant envoyé le message.
JMS_IBM_PUTDATE	Date d'envoi du message.
JMS_IBM_PUTTIME	Heure d'envoi du message.
JMS_IBM_REPORT_COA	Demande de messages de rapport "confirmation à l'arrivée", avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.
JMS_IBM_REPORT_COD	Demande de messages de rapport "confirmation à la livraison", avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.
JMS_IBM_REPORT_DISCARD_MSG	Demande de destruction du message au cas où celui-ci ne peut pas être livré à la destination prévue.
JMS_IBM_REPORT_EXCEPTION	Demande de messages de rapport d'exception, avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.

Tableau 30. Propriétés de Message (suite)	
Nom de la propriété	**Description**
JMS_IBM_REPORT_EXPIRATION	Demande de messages de rapport d'expiration, avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.
JMS_IBM_REPORT_NAN	Demande de messages de rapport de notification d'action négative.
JMS_IBM_REPORT_PAN	Demande de messages de rapport de notification d'action positive.
JMS_IBM_REPORT_PASS_CORREL_ID	Demande que l'identificateur de corrélation de tout message de rapport ou de réponse soit le même que celui du message d'origine.
JMS_IBM_REPORT_PASS_MSG_ID	Demande que l'identificateur de message de tout message de rapport ou de réponse soit le même que celui du message d'origine.
JMS_IBM_RETAIN	La définition de cette propriété indique que le gestionnaire de file d'attente traite un message en tant que publication conservée.
JMS_IBM_SYSTEM_MESSAGEID	Identificateur unique du message dans le bus d'intégration de services. Cette propriété est en lecture seule.
JMSX_APPID	Nom de l'application ayant envoyé le message.
JMSX_DELIVERY_COUNT	Nombre de tentatives de livraison du message.
JMSX_GROUPID	Identificateur du groupe de messages auquel le message appartient.
JMSX_GROUPSEQ	Numéro de séquence du message dans un groupe de messages.
JMSX_USERID	ID utilisateur associé à l'application ayant envoyé le message.

Propriétés JMS_IBM_MQMD*

IBM Message Service Client for .NET permet aux applications client de lire/écrire des zones MQMD à l'aide des API. Il permet également d'accéder aux données de message MQ. Par défaut, l'accès à MQMD est désactivé et doit être activé explicitement par l'application à l'aide des propriétés Destination XMSC_WMQ_MQMD_WRITE_ENABLED et XMSC_WMQ_MQMD_READ_ENABLED. Ces deux propriétés sont indépendantes l'une de l'autre.

Toutes les zones MQMD à l'exception de StrucId et Version sont exposées en tant que propriétés supplémentaires de l'objet Message et portent le préfixe JMS_IBM_MQMD.

Les propriétés JMS_IBM_MQMD* ont priorité sur les autres propriétés JMS_IBM* décrites dans le tableau précédent.

Envoi de messages

Toutes les zones MQMD à l'exception de StrucId et Version sont représentées. Ces propriétés font référence uniquement aux zones MQMD ; lorsqu'une propriété se trouve à la fois dans le MQMD et dans l'en-tête MQRFH2, la version de MQRFH2 n'est ni définie ni extraite. Toutes ces

propriétés peuvent être définies, à l'exception de JMS_IBM_MQMD_BackoutCount. Toute valeur définie JMS_IBM_MQMD_BackoutCount est ignorée.

Si une propriété a une longueur maximale et que vous indiquez une valeur trop longue, celle-ci est tronquée.

Pour certaines propriétés, vous devez également définir la propriété XMSC_WMQ_MQMD_MESSAGE_CONTEXT sur l'objet Destination. L'application doit être exécutée avec les droits appropriés pour que cette propriété soit appliquée. Si vous définissez pas une valeur appropriée pour XMSC_WMQ_MQMD_MESSAGE_CONTEXT, celle-ci est ignorée. Si vous définissez une valeur appropriée pour XMSC_WMQ_MQMD_MESSAGE_CONTEXT, mais que vous ne disposez pas des droits requis pour le gestionnaire de files d'attente, une exception est émise. Les propriétés qui nécessitent des valeurs spécifiques de XMSC_WMQ_MQMD_MESSAGE_CONTEXT sont les suivantes.

Les propriétés suivantes requièrent que XMSC_WMQ_MQMD_MESSAGE_CONTEXT soit défini sur XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT ou XMSC_WMQ_MDCTX_SET_ALL_CONTEXT :

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

Les propriétés suivantes requièrent que XMSC_WMQ_MQMD_MESSAGE_CONTEXT soit défini sur XMSC_WMQ_MDCTX_SET_ALL_CONTEXT :

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

Réception de messages

Toutes ces propriétés sont disponibles sur un message reçu si la propriété XMSC_WMQ_MQMD_READ_ENABLED est définie sur true, quelles que soient les propriétés définies par l'application. Une application ne peut pas modifier les propriétés d'un message reçu, sauf si toutes les propriétés sont d'abord effacées, en fonction de la spécification JMS. Les messages reçus peuvent être réacheminés sans modification des propriétés.

Remarque : Si votre application reçoit un message depuis une destination avec la propriété XMSC_WMQ_MQMD_READ_ENABLED définie sur true, et qu'elle le réachemine vers une destination avec XMSC_WMQ_MQMD_WRITE_ENABLED définie sur true, toutes les valeurs de zone MSMD du message reçu sont copiées dans le message réacheminé. Table des propriétés

Tableau 31. Propriétés de l'objet Message représentant les zones MQMD

Propriété	Description	Tapez
JMS_IBM_MQMD_REPORT	Options des messages de rapport	System.Int32
JMS_IBM_MQMD_MSGTYPE	Type de message	System.Int32
JMS_IBM_MQMD_EXPIRY	Durée de vie du message	System.Int32
JMS_IBM_MQMD_FEEDBACK	Commentaires en retour ou code anomalie	System.Int32
JMS_IBM_MQMD_ENCODING	Codage numérique des données de message	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID	Identificateur de jeu de caractères des données de message	System.Int32

Tableau 31. Propriétés de l'objet Message représentant les zones MQMD (suite)

Propriété	Description	Tapez
JMS_IBM_MQMD_FORMAT	Nom de format des données de message	System.String
JMS_IBM_MQMD_PRIORITY **Remarque :** Toutes valeur affectée à JMS_IBM_MQMD_PRIORITY non comprise entre 0 et 9 enfreint la spécification JMS.	Priorité de message	System.Int32
JMS_IBM_MQMD_PERSISTENCE	Persistance des messages	System.Int32
JMS_IBM_MQMD_MSGID **Remarque :** La spécification JMS indique que l'ID message doit être défini par le fournisseur JMS et qu'il doit être unique ou Null. Si vous affectez une valeur à JMS_IBM_MQMD_MSGID, celle-ci est copiée dans JMSMessageID. Elle n'est alors pas définie par le fournisseur JMS et peut ne pas être unique : cette valeur enfreint la spécification JMS.	Identificateur de message	Tableau d'octets **Remarque :** L'utilisation des propriétés de tableau d'octets sur un message enfreint la spécification JMS.
JMS_IBM_MQMD_CORRELID **Remarque :** Si vous affectez une valeur à JMS_IBM_MQMD_CORRELID qui commence par la chaîne "ID:", cette valeur enfreint la spécification JMS.	Identificateur de corrélation	Tableau d'octets **Remarque :** L'utilisation des propriétés de tableau d'octets sur un message enfreint la spécification JMS.
JMS_IBM_MQMD_BACKOUTCOUNT	Nombre d'annulations	System.Int32
JMS_IBM_MQMD_REPLYTOQ	Nom de la file d'attente de réponses	System.String
JMS_IBM_MQMD_REPLYTOQMGR	Nom du gestionnaire de files d'attente de réponses	System.String
JMS_IBM_MQMD_USERIDENTIFIER	Identificateur utilisateur	System.String
JMS_IBM_MQMD_ACCOUNTINGTOKEN	Jeton de comptabilité	Tableau d'octets **Remarque :** L'utilisation des propriétés de tableau d'octets sur un message enfreint la spécification JMS.
JMS_IBM_MQMD_APPLIDENTITYDATA	Données d'application relatives à l'identité	System.String
JMS_IBM_MQMD_PUTAPPLTYPE	Type de l'application ayant placé le message en file d'attente	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	Nom de l'application ayant placé le message en file d'attente	System.String
JMS_IBM_MQMD_PUTDATE	Date à laquelle le message a été placé en file d'attente	System.String

Tableau 31. Propriétés de l'objet Message représentant les zones MQMD (suite)		
Propriété	**Description**	**Tapez**
JMS_IBM_MQMD_PUTTIME	Heure à laquelle le message a été placé en file d'attente	System.String
JMS_IBM_MQMD_APPLORIGINDATA	Données d'application relatives à l'origine	System.String
JMS_IBM_MQMD_GROUPID	Identificateur de groupe	Tableau d'octets **Remarque :** L'utilisation des propriétés de tableau d'octets sur un message enfreint la spécification JMS.
JMS_IBM_MQMD_MSGSEQNUMBER	Numéro de séquence du message local dans le groupe	System.Int32
JMS_IBM_MQMD_OFFSET	Décalage des données du message physique à partir du début du message logique	System.Int32
JMS_IBM_MQMD_MSGFLAGS	Indicateurs de message	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	Longueur du message d'origine	System.Int32

Voir MQMD pour plus de détails.

Exemples

Cet exemple génère un message placé dans une file d'attente ou une rubrique avec MQMD.UserIdentifier défini sur "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(XMSC_WMQ_MQMD_WRITE_ENABLED,
      XMSC_WMQ_MQMD_WRITE_ENABLED_YES);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(XMSC_WMQ_MQMD_MESSAGE_CONTEXT,
  XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty(JMS_IBM_MQMD_USERIDENTIFIER, "JoeBloggs");

// Send the message
// ...
```

Il est nécessaire de définir XMSC_WMQ_MQMD_MESSAGE_CONTEXT avant JMS_IBM_MQMD_USERIDENTIFIER. Pour plus d'informations sur l'utilisation de XMSC_WMQ_MQMD_MESSAGE_CONTEXT, voir les propriétés de l'objet Message.

De même, vous pouvez extraire le contenu des zones MQMD en définissant XMSC_WMQ_MQMD_READ_ENABLED à true avant la réception d'un message, puis en utilisant les méthodes get du message telles que getStringProperty. Les propriétés reçues sont en lecture seule.

La zone de valeur prend la valeur de la zone MQMD.ApplIdentityData d'un message à partir d'une file d'attente ou d'une rubrique.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(XMSC_WMQ_MQMD_READ_ENABLED, XMSC_WMQ_MQMD_READ_ENABLED_YES);

// Receive a message
// ...

// Get required MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

Propriétés de MessageConsumer

Présentation des propriétés de l'objet MessageConsumer, avec des liens vers des informations de référence plus détaillées.

Tableau 32. Propriétés de MessageConsumer

Nom de la propriété	Description
XMSC_IS_SUBSCRIPTION_MULTICAST	Indique si les messages sont distribués au consommateur de message à l'aide de WebSphere MQ Multicast Transport. Cette propriété est en lecture seule.
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	Indique si les messages sont distribués au consommateur de message à l'aide de WebSphere MQ Multicast Transport avec une qualité de service fiable. Cette propriété est en lecture seule.

Pour plus de détails, voir Propriétés .NET de IMessageConsumer.

Propriétés de MessageProducer

Présentation des propriétés de l'objet MessageProducer, avec des liens vers des informations de référence plus détaillées.

Voir .Propriétés NET de IMessageProducer pour plus de détails.

Propriétés de Session

Présentation des propriétés de l'objet Session, avec des liens vers des informations de référence plus détaillées.

Voir .Propriétés NET d'ISession pour plus de détails.

Définitions de propriété

Cette section fournit une définition de chaque propriété d'objet.

Chaque définition de propriété inclut les informations suivantes :

- Le type de données de la propriété
- Les types d'objet de la propriété
- Pour une propriété de Destination, le nom qui peut être utilisé dans un identificateur URI
- Une description plus détaillée de la propriété
- Les valeurs admises de la propriété

- La valeur par défaut de la propriété

Les propriétés dont le nom commence par un des préfixes suivants sont appropriées uniquement pour le type de connexion spécifié :

XMSC_RTT
Les propriétés sont appropriées uniquement pour une connexion en temps réelle à un courtier. Les noms de propriété sont définis en tant que constantes nommées dans le fichier d'en-tête `xmsc_rtt.h`.

XMSC_WMQ
Les propriétés sont appropriées uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente WebSphere MQ. Les noms de propriété sont définis en tant que constantes nommées dans le fichier d'en-tête `xmsc_wmq.h`.

XMSC_WPM
Les propriétés sont appropriées uniquement lorsqu'une application se connecte à un bus d'intégration de services WebSphere. Les noms de propriété sont définis en tant que constantes nommées dans le fichier d'en-tête `xmsc_wpm.h`.

Sauf indication contraire dans leur définition, les propriétés restantes sont pertinentes pour tous les types de connexion. Les noms de propriété sont définis en tant que constantes nommées dans le fichier d'en-tête `xmsc.h`. Les propriétés dont le nom commence par le préfixe JMSX sont les propriétés de message définies par JMS, et les propriétés dont le nom commence par le préfixe JMS_IBM sont propriétés de message définies par IBM. Pour plus d'informations sur les propriétés des messages, voir «Propriétés d'un message XMS», à la page 72.

Sauf indication contraire dans sa définition, chaque propriété est pertinente dans les domaines point-à-point et Publier/S'abonner.

Une application peut extraire et définir la valeur de toute propriété, sauf si celle-ci est désignée comme étant en lecture seule.

JMS_IBM_CHARACTER_SET

Type de données :
System.Int32

Propriété de :
Message

Identificateur (CCSID) du jeu de caractères codés, ou page de codes, dans lequel se trouvent les chaînes de données de type caractères dans le corps du message lorsque le client XMS transmet le message à la destination prévue. Dans XMS, cette propriété a une valeur numérique mappée vers le CCSID. Toutefois, cette propriété est basée sur une propriété JMS qui contient une valeur de type chaîne et mappe vers le jeu de caractères Java qui représente ce CCSID numérique. Cette propriété se substitue à tout CCSID spécifié pour la destination par la propriété XMSC_WMQ_CCSID.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_ENCODING

Type de données :
System.Int32

Propriété de :
Message

Mode de représentation des données numériques dans le corps du message lorsque le client XMS transmet le message à la destination prévue. Cette propriété se substitue à tout codage spécifié pour la destination par la propriété XMSC_WMQ_ENCODING. La propriété spécifie la représentation d'entiers binaires, d'entiers décimaux condensés et de nombres à virgule flottante.

Les valeurs valides de la propriété sont identiques à celles qui peuvent être spécifiées dans la zone **Encoding** d'un descripteur de message.

Une application peut utiliser les constantes nommées suivantes pour définir la propriété :

Constante nommée	Explication
MQENC_INTEGER_NORMAL	Codage d'entier normal
MQENC_INTEGER_REVERSED	Codage d'entier inversé
MQENC_DECIMAL_NORMAL	Codage décimal condensé normal
MQENC_DECIMAL_REVERSED	Codage décimal condensé inversé
MQENC_FLOAT_IEEE_NORMAL	Codage à virgule flottante IEEE normal
MQENC_FLOAT_IEEE_REVERSED	Codage à virgule flottante IEEE inversé
MQENC_FLOAT_S390	z/OS Codage à virgule flottante à architecture
MQENC_NATIVE	Codage machine natif

Pour former une valeur pour la propriété, l'application peut ajouter trois des constantes suivantes :

- Une constante dont le nom commence par MQENC_INTEGER, pour spécifier la représentation d'entiers binaires
- Une constante dont le nom commence par MQENC_DECIMAL, pour spécifier la représentation d'entiers décimaux condensés
- Une constante dont le nom commence par MQENC_FLOAT, pour spécifier la représentation de nombres à virgule flottante

L'application peut également définir la propriété à MQENC_NATIVE, dont la valeur dépend de l'environnement.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_EXCEPTIONMESSAGE

Type de données :
String

Propriété de :
Message

Texte décrivant la raison pour laquelle le message a été envoyé à la destination d'exception. Cette propriété est en lecture seule.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un bus d'intégration de services et reçoit un message d'une destination d'exception.

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

Type de données :
String

Propriété de :
Message

Nom de la destination du message avant que celui-ci soit envoyé à la destination d'exception.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un bus d'intégration de services et reçoit un message d'une destination d'exception.

JMS_IBM_EXCEPTIONREASON

Type de données :
System.Int32

Propriété de :
Message

Code anomalie indiquant la raison pour laquelle le message a été envoyé à la destination d'exception.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un bus d'intégration de services et reçoit un message d'une destination d'exception.

JMS_IBM_EXCEPTIONTIMESTAMP

Type de données :
System.Int64

Propriété de :
Message

Heure à laquelle le message a été envoyé à la destination d'exception.

L'heure est exprimée en millisecondes, à partir du 1er janvier 1970 00:00:00 GMT.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un bus d'intégration de services et reçoit un message d'une destination d'exception.

JMS_IBM_FEEDBACK

Type de données :
System.Int32

Propriété de :
Message

Code qui indique la nature d'un message de rapport.

Les valeurs valides de la propriété sont les codes retour et les codes raison qui peuvent être spécifiés dans la zone **Feedback** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

JMS_IBM_FORMAT

Type de données :
String

Propriété de :
Message

Nature des données d'application dans le message.

Les valeurs valides de la propriété sont identiques à celles qui peuvent être spécifiées dans la zone **Format** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_LAST_MSG_IN_GROUP

Type de données :
System.Boolean

Propriété de :
Message

Indique si le message est le dernier d'un groupe de messages.

Définissez la propriété à true si le message est le dernier d'un groupe de messages. Dans le cas contraire, définissez la propriété à false ou ne la définissez pas. Par défaut, la propriété n'est pas définie.

La valeur true correspond à l'indicateur d'état MQMF_LAST_MSG_IN_GROUP, qui peut être spécifié dans la zone **MsgFlags** d'un descripteur de message.

Cette propriété est ignorée dans le domaine Publier/S'abonner et n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_MSGTYPE

Type de données :
System.Int32

Propriété de :
Message

Type du message.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQMT_DATAGRAM	Le message ne requiert pas de réponse.
MQMT_REQUEST	Le message requiert une réponse.
MQMT_REPLY	Le message est un message de réponse.
MQMT_REPORT	Le message est un message de rapport.

Ces valeurs correspondent aux types de message qui peuvent être spécifiés dans la zone **MsgType** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_PUTAPPLTYPE

Type de données :
System.Int32

Propriété de :
Message

Type de l'application ayant envoyé le message.

Les valeurs admises sont les types d'application qui peuvent être spécifiés dans la zone **PutApplType** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_PUTDATE

Type de données :
String

Propriété de :
Message

Date d'envoi du message.

Les valeurs valides de la propriété sont identiques à celles qui peuvent être spécifiées dans la zone **PutDate** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_PUTTIME

Type de données :
String

Propriété de :
Message

Heure d'envoi du message.

Les valeurs valides de la propriété sont identiques à celles qui peuvent être spécifiées dans la zone **PutTime** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas pertinente lorsqu'une application se connecte à un bus d'intégration de services.

JMS_IBM_REPORT_COA

Type de données :
System.Int32

Propriété de :
Message

Demande de messages de rapport "confirmation à l'arrivée", avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQRO_COA	Demande de messages de rapport "confirmation à l'arrivée", sans données d'application du message d'origine incluses dans un message de rapport.
MQRO_COA_WITH_DATA	Demande de messages de rapport "confirmation à l'arrivée", avec les 100 premiers octets de données d'application du message d'origine incluses dans un message de rapport.
MQRO_COA_WITH_FULL_DATA	Demande de messages de rapport "confirmation à l'arrivée", avec toutes les données d'application du message d'origine incluses dans un message de rapport.

Ces valeurs correspondent aux options de rapport qui peuvent être spécifiées dans la zone **Report** d'un descripteur de message. Pour plus d'informations sur ces options, voir Rapport (MQLONG).

Par défaut, la propriété n'est pas définie.

JMS_IBM_REPORT_COD

Type de données :
System.Int32

Propriété de :
Message

Demande de messages de rapport "confirmation à la livraison", avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQRO_COD	Demande de messages de rapport "confirmation à la livraison", sans données d'application du message d'origine incluses dans un message de rapport.
MQRO_COD_WITH_DATA	Demande de messages de rapport "confirmation à la livraison", avec les 100 premiers octets de données d'application du message d'origine incluses dans un message de rapport.
MQRO_COD_WITH_FULL_DATA	Demande de messages de rapport "confirmation à la livraison", avec toutes les données d'application du message d'origine incluses dans un message de rapport.

Ces valeurs correspondent aux options de rapport qui peuvent être spécifiées dans la zone **Report** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

JMS_IBM_REPORT_DISCARD_MSG

Type de données :
System.Int32

Propriété de :
Message

Demande de destruction du message au cas où celui-ci ne peut pas être livré à la destination prévue.

Définissez la propriété à MQRO_DISCARD_MSG pour demander la destruction du message au cas où celui-ci ne peut être livré à la destination prévue. Si vous souhaitez que le message soit placé dans une file d'attente de messages non livrés ou envoyé à une destination d'exception, ne définissez pas cette propriété. Par défaut, la propriété n'est pas définie.

La valeur MQRO_DISCARD_MSG correspond à une option de rapport qui peut être spécifiée dans la zone **Report** d'un descripteur de message.

JMS_IBM_REPORT_EXCEPTION

Type de données :
System.Int32

Propriété de :
Message

Demande de messages de rapport d'exception, avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQRO_EXCEPTION	Demande de messages de rapport d'exception, sans données d'application du message d'origine incluses dans un message de rapport.
MQRO_EXCEPTION_WITH_DATA	Demande de messages de rapport d'exception, avec les 100 premiers octets de données d'application du message d'origine incluses dans un message de rapport.
MQRO_EXCEPTION_WITH_FULL_DATA	Demande de messages de rapport d'exception, avec toutes les données d'application du message d'origine incluses dans un message de rapport.

Ces valeurs correspondent aux options de rapport qui peuvent être spécifiées dans la zone **Report** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

JMS_IBM_REPORT_EXPIRATION

Type de données :
System.Int32

Propriété de :
Message

Demande de messages de rapport d'expiration, avec indication de la quantité de données d'application du message d'origine devant être incluse dans un message de rapport.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQRO_EXPIRATION	Demande de messages de rapport d'expiration, sans données d'application du message d'origine incluses dans un message de rapport.
MQRO_EXPIRATION_WITH_DATA	Demande de messages de rapport d'expiration, avec les 100 premiers octets de données d'application du message d'origine incluses dans un message de rapport.
MQRO_EXPIRATION_WITH_FULL_DATA	Demande de messages de rapport d'expiration, avec toutes les données d'application du message d'origine incluses dans un message de rapport.

Ces valeurs correspondent aux options de rapport qui peuvent être spécifiées dans la zone **Report** d'un descripteur de message.

Par défaut, la propriété n'est pas définie.

JMS_IBM_REPORT_NAN

Type de données :
System.Int32

Propriété de :
Message

Demande de messages de rapport de notification d'action négative.

Définissez la propriété à MQRO_NAN pour demander des messages de rapport de notification d'action négative. Si vous ne voulez pas de des messages de rapport de notification d'action négative, ne définissez pas cette propriété. Par défaut, la propriété n'est pas définie.

La valeur MQRO_NAN correspond à une option de rapport qui peut être spécifiée dans la zone **Report** d'un descripteur de message.

JMS_IBM_REPORT_PAN

Type de données :
System.Int32

Propriété de :
Message

Demande de messages de rapport de notification d'action positive.

Définissez la propriété à MQRO_PAN pour demander des messages de rapport de notification d'action positive. Si vous ne voulez pas de des messages de rapport de notification d'action positive, ne définissez pas cette propriété. Par défaut, la propriété n'est pas définie.

La valeur MQRO_PAN correspond à une option de rapport qui peut être spécifiée dans la zone **Report** d'un descripteur de message.

JMS_IBM_REPORT_PASS_CORREL_ID

Type de données :
System.Int32

Propriété de :
Message

Demande que l'identificateur de corrélation de tout message de rapport ou de réponse soit le même que celui du message d'origine.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQRO_PASS_CORREL_ID	Demande selon laquelle l'identificateur de corrélation de tout message de rapport ou de réponse doit être le même que celui du message d'origine.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Demande que l'identificateur de corrélation de tout message de rapport ou de réponse soit le même que l'identificateur de message du message d'origine.

Ces valeurs correspondent aux options de rapport qui peuvent être spécifiées dans la zone **Report** d'un descripteur de message.

La valeur par défaut de la propriété est MQRO_COPY_MSG_ID_TO_CORREL_ID.

JMS_IBM_REPORT_PASS_MSG_ID

Type de données :
System.Int32

Propriété de :
Message

Demande que l'identificateur de message de tout message de rapport ou de réponse soit le même que celui du message d'origine.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
MQRO_PASS_MSG_ID	Demande selon laquelle l'identificateur de message de tout message de rapport ou de réponse doit être le même que celui du message d'origine.
MQRO_NEW_MSG_ID	Demande qu'un nouvel identificateur de message soit généré pour chaque message de rapport ou de réponse.

Ces valeurs correspondent aux options de rapport qui peuvent être spécifiées dans la zone Rapport d'un descripteur de message.

La valeur par défaut de la propriété est MQRO_NEW_MSG_ID.

JMS_IBM_RETAIN

Type de données :
System.Int32

Propriété de :
Message

La définition de cette propriété indique que le gestionnaire de file d'attente traite un message en tant que publication conservée. Lorsqu'un abonné reçoit des messages provenant de rubriques, il peut recevoir des messages supplémentaires immédiatement après l'abonnement, en plus de ceux reçus dans les versions précédentes. Ces messages sont les publications conservées (facultatives) pour les rubriques concernées par l'abonnement. Pour chaque rubrique correspondant à l'abonnement, les publications conservées sont disponibles pour livraison au consommateur de message abonné.

RETAIN_PUBLICATION est la seule valeur admise pour cette propriété. Par défaut, cette propriété n'est pas définie.

Remarque : Cette propriété est pertinente uniquement dans le domaine publication/abonnement.

JMS_IBM_SYSTEM_MESSAGEID

Type de données :
String

Propriété de :
Message

Identificateur unique du message dans le bus d'intégration de services. Cette propriété est en lecture seule.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un bus d'intégration de services.

JMSX_APPID

Type de données :
String

Propriété de :
Message

Nom de l'application ayant envoyé le message.

Il s'agit de la propriété JMS définie avec le nom JMS JMSXAppID. Pour plus d'informations sur la propriété, voir *Java Message Service Specification, Version 1.1*.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas valide pour une connexion en temps réel à un courtier.

JMSX_DELIVERY_COUNT

Type de données :
System.Int32

Propriété de :
Message

Nombre de tentatives de livraison du message.

Il s'agit de la propriété JMS définie avec le nom JMS JMSXDeliveryCount. Pour plus d'informations sur la propriété, voir *Java Message Service Specification, Version 1.1*.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas valide pour une connexion en temps réel à un courtier.

JMSX_GROUPID

Type de données :
String

Propriété de :
Message

Identificateur du groupe de messages auquel le message appartient.

Il s'agit de la propriété JMS définie avec le nom JMS JMSXGroupID. Pour plus d'informations sur la propriété, voir *Java Message Service Specification, Version 1.1*.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas valide pour une connexion en temps réel à un courtier.

JMSX_GROUPSEQ

Type de données :
System.Int32

Propriété de :
Message

Numéro de séquence du message dans un groupe de messages.

Il s'agit de la propriété JMS définie avec le nom JMS JMSXGroupSeq. Pour plus d'informations sur la propriété, voir *Java Message Service Specification, Version 1.1*.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas valide pour une connexion en temps réel à un courtier.

JMSX_USERID

Type de données :
String

Propriété de :
Message

ID utilisateur associé à l'application ayant envoyé le message.

Il s'agit de la propriété JMS définie avec le nom JMS JMSXUserID. Pour plus d'informations sur la propriété, voir *Java Message Service Specification, Version 1.1*.

Par défaut, la propriété n'est pas définie.

Cette propriété n'est pas valide pour une connexion en temps réel à un courtier.

XMSC_ASYNC_EXCEPTIONS

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: ASYNCEXCEPTION

Nom abrégé de l'outil d'administration JMS: AEX

Cette propriété détermine si XMS informe un programme d'écoute des exceptions uniquement lorsqu'une connexion est interrompue, ou également lorsqu'une exception est émise de façon asynchrone sur un appel d'API XMS. Cette propriété s'applique à toutes les connexions créées à partir de la fabrique de connexions et pour lesquelles un programme d'écoute des exceptions est enregistré.

Les valeurs valides de cette propriété sont les suivantes :

XMSC_ASYNC_EXCEPTIONS_ALL
Toutes les exceptions détectées de façon asynchrone, en dehors de la portée d'un appel d'API synchrone, et toutes les exceptions de connexion interrompue sont envoyées au programme d'écoute des exceptions.

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN
Seules les exceptions signalant une connexion interrompue sont envoyées au programme d'écoute des exceptions. Toutes les autres exceptions se produisant au cours d'un traitement asynchrone ne sont pas rapportées au programme d'écoute des exceptions ; l'application n'est donc pas informée de ces exceptions.

Par défaut, cette propriété est définie sur XMSC_ASYNC_EXCEPTIONS_ALL.

XMSC_CLIENT_ID

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: CLIENTID

Nom abrégé de l'outil d'administration JMS: CID

Identificateur de client pour une connexion.

Un identificateur de client est utilisé uniquement pour la prise en charge des abonnements durables dans le domaine Publier/S'abonner ; il est ignoré dans le domaine point-à-point. Pour plus d'informations sur le paramétrage des identificateurs de client, voir«Objets ConnectionFactories et Connection», à la page 21.

Cette propriété n'est pas pertinente pour une connexion en temps réel à un courtier.

XMSC_CONNECTION_TYPE

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Type de serveur de messagerie auquel une application se connecte.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_CT_RTT	Connexion en temps réel à un courtier.
XMSC_CT_WMQ	Connexion à un gestionnaire de files d'attente WebSphere MQ.
XMSC_CT_WPM	Connexion à un bus d'intégration de services WebSphere.

Par défaut, la propriété n'est pas définie.

XMSC_DELIVERY_MODE

Type de données :
System.Int32

Propriété de :
Destination

Nom utilisé dans un URI :

`persistence` (pour une destination WebSphere MQ)
`deliveryMode` (pour une destination de fournisseur de messagerie par défaut WebSphere)

Objets applicables:
Nom long de l'outil d'administration JMS: PERSISTENCE

Nom abrégé de l'outil d'administration JMS: PER

Mode de livraison des messages envoyés à la destination.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_DELIVERY_NOT_PERSISTENT	Un message envoyé à la destination est non permanent. Le mode de livraison par défaut de l'expéditeur de message, ou tout mode de livraison spécifié sur l'appel Send, est ignoré. Si la destination est une file d'attente WebSphere MQ, la valeur de l'attribut de file d'attente *DefPersistence* est également ignorée.
XMSC_DELIVERY_PERSISTENT	Un message envoyé à la destination est permanent. Le mode de livraison par défaut de l'expéditeur de message, ou tout mode de livraison spécifié sur l'appel Send, est ignoré. Si la destination est une file d'attente WebSphere MQ, la valeur de l'attribut de file d'attente *DefPersistence* est également ignorée.
XMSC_DELIVERY_AS_APP	Un message envoyé à la destination dont le mode de livraison est spécifié sur l'appel Send. Si l'appel Send ne spécifie pas de mode de livraison, le mode de livraison par défaut de l'expéditeur de message est utilisé. Si la destination est une file d'attente WebSphere MQ, la valeur de l'attribut de file d'attente *DefPersistence* est ignorée.
XMSC_DELIVERY_AS_DEST	Si la destination est une file d'attente WebSphere MQ, le mode de livraison d'un message mis en file d'attente est spécifié par la valeur de l'attribut de file d'attente *DefPersistence*. Le mode de livraison par défaut de l'expéditeur de message, ou tout mode de livraison spécifié sur l'appel Send, est ignoré. Si la destination n'est pas une file d'attente WebSphere MQ, la signification est identique à celle de XMSC_DELIVERY_AS_APP.

La valeur par défaut est XMSC_DELIVERY_AS_APP.

XMSC_IC_PROVIDER_URL

Type de données :
String

Propriété de :
InitialContext

Utilisé pour localiser le répertoire de désignation JNDI de sorte que le service de dénomination COS n'ait pas besoin d'être sur le même serveur que le service Web.

XMSC_IC_SECURITY_AUTHENTICATION

Type de données :
String

Propriété de :
InitialContext

Basé sur l'interface de contexte Java SECURITY_AUTHENTICATION. Cette propriété est applicable uniquement au contexte d'affectation de nom de classe de service.

XMSC_IC_SECURITY_CREDENTIALS

Type de données :
String

Propriété de :
InitialContext

Basé sur l'interface de contexte Java SECURITY_CREDENTIALS. Cette propriété est applicable uniquement au contexte d'affectation de nom de classe de service.

XMSC_IC_SECURITY_PRINCIPAL

Type de données :
String

Propriété de :
InitialContext

Basé sur l'interface de contexte Java SECURITY_PRINCIPAL. Cette propriété est applicable uniquement au contexte d'affectation de nom de classe de service.

XMSC_IC_SECURITY_PROTOCOL

Type de données :
String

Propriété de :
InitialContext

Basé sur l'interface de contexte Java SECURITY_PROTOCOL . Cette propriété s'applique uniquement au contexte de nommage COS.

XMSC_IC_URL

Type de données :
String

Propriété de :
InitialContext

Pour les contextes LDAP et FileSystem, adresse du référentiel contenant les objets gérés.

Pour les contextes d'affectation de nom de classe de service, adresse du service Web qui recherche les objets dans le répertoire.

XMSC_IS_SUBSCRIPTION_MULTICAST

Type de données :
System.Boolean

Propriété de :
MessageConsumer

Indique si les messages sont distribués au consommateur de message à l'aide de WebSphere MQ Multicast Transport. Cette propriété est en lecture seule.

La valeur de la propriété est true si les messages sont distribués au consommateur de message via WebSphere MQ Multicast Transport. Dans le cas contraire, la valeur est false.

Cette propriété est pertinente uniquement pour une connexion en temps réel à un courtier.

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

Type de données :
System.Boolean

Propriété de :
MessageConsumer

Indique si les messages sont distribués au consommateur de message à l'aide de WebSphere MQ Multicast Transport avec une qualité de service fiable. Cette propriété est en lecture seule.

La valeur de la propriété est true si les messages sont distribués au consommateur de message via WebSphere MQ Multicast Transport avec une qualité de service fiable. Dans le cas contraire, la valeur est false.

Cette propriété est pertinente uniquement pour une connexion en temps réel à un courtier.

XMSC_JMS_MAJOR_VERSION

Type de données :
System.Int32

Propriété de :
ConnectionMetaData

Numéro de version principale de la spécification JMS sur laquelle repose XMS . Cette propriété est en lecture seule.

XMSC_JMS_MINOR_VERSION

Type de données :
System.Int32

Propriété de :
ConnectionMetaData

Numéro de version secondaire de la spécification JMS sur laquelle repose XMS . Cette propriété est en lecture seule.

XMSC_JMS_VERSION

Type de données :
String

Propriété de :
ConnectionMetaData

Identificateur de version de la spécification JMS sur laquelle est basée XMS . Cette propriété est en lecture seule.

XMSC_MAJOR_VERSION

Type de données :
System.Int32

Propriété de :
ConnectionMetaData

Numéro de version du client XMS . Cette propriété est en lecture seule.

XMSC_MINOR_VERSION

Type de données :
System.Int32

Propriété de :
ConnectionMetaData

Numéro d'édition du client XMS . Cette propriété est en lecture seule.

XMSC_PASSWORD

Type de données :
Tableau d'octets

Propriété de :
ConnectionFactory

Mot de passe pouvant être utilisé pour authentifier l'application lorsqu'elle tente de se connecter à un serveur de messagerie. Le mot de passe est utilisé avec la propriété XMSC_USERID.

Par défaut, la propriété n'est pas définie.

Si vous vous connectez à WebSphere MQ sur Multiplateformeset que vous définissez la propriété XMSC_USERID de la fabrique de connexions, elle doit correspondre au **userid** de l'utilisateur connecté. Si vous ne définissez pas ces propriétés, le gestionnaire de files d'attente utilise par défaut le **userid** de l'utilisateur connecté. Si vous avez besoin d'authentifications de niveau connexion pour des utilisateurs individuels, vous pouvez écrire un exit d'authentification de client, configuré dans WebSphere MQ. Vous trouverez des informations complémentaires sur la création d'un exit d'authentification de client dans la rubrique Authentication du manuel WebSphere MQ Clients.

Pour authentifier l'utilisateur lors de la connexion à WebSphere MQ sur z/OS, vous devez utiliser un exit de sécurité.

XMSC_PRIORITY

Type de données :
System.Int32

Propriété de :
Destination

Nom utilisé dans un URI :
priorité

Priorité des messages envoyés à la destination.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
Un entier, compris entre 0, priorité la plus faible, et 9, priorité la plus élevée	Un message envoyé à la destination a la priorité spécifiée. La priorité par défaut de l'expéditeur de message, ou toute priorité spécifiée sur l'appel Send, est ignorée. Si la destination est une file d'attente WebSphere MQ, la valeur de l'attribut de file d'attente **DefPriority** est également ignorée.
XMSC_PRIORITY_AS_APP	Un message envoyé à la destination a la priorité spécifiée sur l'appel Send. Si l'appel Send ne spécifie aucun priorité, la priorité par défaut de l'expéditeur de message est utilisée. Si la destination est une file d'attente WebSphere MQ, la valeur de l'attribut de file d'attente **DefPriority** est ignorée.
XMSC_PRIORITY_AS_DEST	Si la destination est une file d'attente WebSphere MQ, un message placé en file d'attente a le niveau de priorité indiqué par la valeur de l'attribut de file d'attente **DefPriority**. La priorité par défaut de l'expéditeur de message, ou toute priorité spécifiée sur l'appel Send, est ignorée. Si la destination n'est pas une file d'attente WebSphere MQ, la signification est identique à celle de XMSC_PRIORITY_AS_APP.

La valeur par défaut est XMSC_PRIORITY_AS_APP.

WebSphere MQ Real-Time Transport et WebSphere MQ Multicast Transport n'effectuent aucune action basée sur la priorité d'un message.

XMSC_PROVIDER_NAME

Type de données :
String

Propriété de :
ConnectionMetaData

Fournisseur du client XMS . Cette propriété est en lecture seule.

XMSC_RTT_BROKER_PING_INTERVAL

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Intervalle de temps, en millisecondes, après lequel XMS .NET vérifie la connexion à un serveur de messagerie en temps réel pour détecter une activité. Si aucune activité n'est détectée, le client initie une commande ping ; la connexion est fermée si aucune réponse à la commande ping n'est détectée.

La valeur par défaut de la propriété est 30000.

XMSC_RTT_CONNECTION_PROTOCOL

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Protocole de communication utilisé pour une connexion en temps réel à un courtier.

La valeur de la propriété doit être XMSC_RTT_CP_TCP, ce qui indique une connexion en temps réel à un courtier via TCP/IP. La valeur par défaut est XMSC_RTT_CP_TCP.

XMSC_RTT_HOST_NAME

Type de données :
String

Propriété de :
ConnectionFactory

Nom d'hôte ou adresse IP du système sur lequel un courtier s'exécute.

Cette propriété est utilisée avec la propriété XMSC_RTT_PORT pour identifier le courtier.

Par défaut, la propriété n'est pas définie.

XMSC_RTT_LOCAL_ADDRESS

Type de données :
String

Propriété de :
ConnectionFactory

Nom d'hôte ou adresse IP de l'interface de réseau local à utiliser pour une connexion en temps réel à un courtier.

Cette propriété est utile uniquement si le système sur lequel l'application est exécutée dispose d'au moins deux interfaces réseau et que vous devez pouvoir spécifier l'interface qui doit être utilisée pour une connexion en temps réel. Si le système dispose d'une seule interface réseau, seule cette interface peut

être utilisée. Si le système dispose d'au moins deux interfaces réseau et que la propriété n'est pas définie, l'interface est sélectionnée au hasard.

Par défaut, la propriété n'est pas définie.

XMSC_RTT_MULTICAST

Type de données :
System.Int32

Propriété de :
ConnectionFactory et Destination

Nom utilisé dans un URI :
multicast

Paramètre de multidiffusion pour une fabrique de connexions ou une destination. Seule une destination représentant une rubrique peut avoir cette propriété.

Une application utilise cette propriété pour activer la multidiffusion en association avec une connexion en temps réel à un courtier et, si la multidiffusion est activée, pour spécifier la manière précise dont la multidiffusion est utilisée pour distribuer les messages depuis un courtier vers un consommateur de messages. La propriété n'a pas d'effet sur la manière dont un expéditeur de message envoie des messages au courtier.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_RTT_MULTICAST_DISABLED	Les messages ne sont pas distribués à un consommateur de message à l'aide de WebSphere MQ Multicast Transport. Il s'agit de la valeur par défaut pour un objet ConnectionFactory.
XMSC_RTT_MULTICAST_ASCF	Les messages sont distribués à un consommateur de message en fonction du paramètre de multidiffusion défini pour la fabrique de connexions associée au consommateur de message. Le paramètre de multidiffusion défini pour la fabrique de connexions est déterminé au moment de la création de la connexion. Cette valeur est admise uniquement pour un objet Destination, dont elle est la valeur par défaut.
XMSC_RTT_MULTICAST_ENABLED	Si la rubrique est configurée pour la multidiffusion dans le courtier, les messages sont distribués à un consommateur de message à l'aide de WebSphere MQ Multicast Transport. Une qualité de service fiable est utilisée si la rubrique est configurée pour une multidiffusion fiable.
XMSC_RTT_MULTICAST_RELIABLE	Si la rubrique est configurée pour la multidiffusion fiable dans le courtier, les messages sont distribués à un consommateur de message à l'aide de WebSphere MQ Multicast Transport avec une qualité de service fiable. Si la rubrique n'est pas configurée pour la multidiffusion fiable, vous ne pouvez pas créer de consommateur de message pour la rubrique.

Valeur admise	Explication
XMSC_RTT_MULTICAST_NOT_RELIABLE	Si la rubrique est configurée pour la multidiffusion dans le courtier, les messages sont distribués à un consommateur de message à l'aide de WebSphere MQ Multicast Transport. Une qualité de service fiable n'est pas utilisée, même si la rubrique est configurée pour une multidiffusion fiable.

XMSC_RTT_PORT

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Numéro du port sur lequel un courtier écoute les demandes entrantes. Sur le courtier, vous devez configurer un noeud de traitement de message Real-timeInput ou Real-timeOptimizedFlow pour écouter sur ce port.

Cette propriété est utilisée avec la propriété XMSC_RTT_HOST_NAME pour identifier le courtier.

La valeur par défaut de la propriété est XMSC_RTT_DEFAULT_PORT, ou 1506.

XMSC_TIME_TO_LIVE

Type de données :
System.Int32

Propriété de :
Destination

Nom utilisé dans un URI :

expiry (pour une destination WebSphere MQ)
timeToLive (pour une destination fournisseur de messagerie par défaut WebSphere)

Durée de vie des messages envoyés à la destination.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
0	Un message envoyé à la destination n'expire jamais.
Entier positif	Un message envoyé à la destination a la durée de vie spécifiée (en millisecondes). La durée de vie par défaut de l'expéditeur de message ou toute durée de vie spécifiée sur l'appel Send est ignorée.
XMSC_TIME_TO_LIVE_AS_APP	Un message envoyé à la destination a la durée de vie spécifiée sur l'appel Send. Si l'appel Send ne spécifie pas de durée de vie, la valeur par défaut de l'expéditeur de message est utilisée.

La valeur par défaut est XMSC_TIME_TO_LIVE_AS_APP.

XMSC_USERID

Type de données :
String

Propriété de :
ConnectionFactory

ID utilisateur pouvant être utilisé pour authentifier l'application lorsqu'elle tente de se connecter à un serveur de messagerie. L'ID utilisateur est utilisé avec la propriété XMSC_PASSWORD.

Par défaut, la propriété n'est pas définie.

Multi Si vous vous connectez à IBM MQ for Multiplatformset que vous définissez la propriété XMSC_USERID de la fabrique de connexions, elle doit correspondre au **userid** de l'utilisateur connecté. Si vous ne définissez pas ces propriétés, le gestionnaire de files d'attente utilise par défaut le **userid** de l'utilisateur connecté. Si vous avez besoin d'une authentification supplémentaire au niveau de la connexion d'utilisateurs individuels, vous pouvez écrire un exit d'authentification de client configuré dans IBM MQ.

z/OS Pour authentifier l'utilisateur lors de la connexion à IBM MQ for z/OS, vous devez utiliser un exit de sécurité.

XMSC_VERSION

Type de données :
String

Propriété de :
ConnectionMetaData

Identificateur de version du client XMS . Cette propriété est en lecture seule.

XMSC_WMQ_BROKER_CONTROLQ

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la file d'attente de contrôle utilisée par un courtier.

La valeur par défaut de la propriété est SYSTEM.BROKER.CONTROL.QUEUE.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WMQ_BROKER_PUBQ

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la file d'attente contrôlée par un courtier lorsque des applications envoient les messages qu'elles publient.

La valeur par défaut de la propriété est SYSTEM.BROKER.DEFAULT.STREAM.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WMQ_BROKER_QMGR

Type de données :
String

Propriété de :
ConnectionFactory

Nom du gestionnaire de files d'attente auquel un courtier est connecté.

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WMQ_BROKER_SUBQ

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la file d'attente de souscription pour un consommateur de message non durable.

Le nom de la file d'attente de souscription doit commencer par les caractères suivants :

SYSTEM.JMS.ND.

Si vous voulez que tous les consommateurs de message non durables partagent une file d'attente de souscription, spécifiez le nom complet de la file d'attente partagée. Une file d'attente du nom spécifié doit exister avant qu'une application puisse créer un consommateur de message non durable.

Si vous souhaitez que chaque consommateur de message non durable puisse extraire des messages de sa propre file d'attente de souscription exclusive, indiquez un nom de file d'attente qui se termine par un astérisque (*). Ensuite, lorsqu'une application crée un consommateur de message non durable, le client XMS crée une file d'attente dynamique à l'usage exclusif du consommateur de message. Le client XMS utilise la valeur de la propriété pour définir le contenu de la zone **DynamicQName** dans le descripteur d'objet utilisé pour créer la file d'attente dynamique.

La valeur par défaut de la propriété est SYSTEM.JMS.ND.SUBSCRIBER.QUEUE, qui signifie que XMS utilise l'approche de file d'attente partagée par défaut.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WMQ_BROKER_VERSION

Type de données :
System.Int32

Propriété de :
ConnectionFactory et Destination

Nom utilisé dans un URI :
brokerVersion

Type de courtier utilisé par l'application pour une connexion ou pour la destination. Seule une destination représentant une rubrique peut avoir cette propriété.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WMQ_BROKER_V1	L'application utilise un courtier WebSphere MQ Publier/S'abonner. L'application peut aussi utiliser cette valeur si vous migrez de WebSphere MQ Publier/S'abonner vers WebSphere Message Broker sans changer l'application.
XMSC_WMQ_BROKER_V2	L'application utilise un courtier de IBM Integration Bus.
XMSC_WMQ_BROKER_UNSPECIFIED	Lorsque le courtier est migré, définissez cette propriété de sorte que les en-têtes RFH2 ne soient plus utilisés. Après la migration, cette propriété n'est plus significative.

La valeur par défaut pour une fabrique de connexions est XMSC_WMQ_BROKER_UNSPECIFIED, mais par défaut, la propriété n'est pas définie pour une destination. La définition de la propriété pour une destination remplace toute valeur spécifiée par la propriété de fabrique de connexions.

XMSC_WMQ_CCDTURL

Type de données :
System.String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: CCDTURL

Nom abrégé de l'outil d'administration JMS: CCDT

Adresse URL qui identifie le nom et l'emplacement du fichier contenant la table de définition de canal du client et qui spécifie le mode d'accès au fichier.

Par défaut, cette propriété n'est pas définie.

XMSC_WMQ_CCSID

Type de données :
System.Int32

Propriété de :
Destination

Nom utilisé dans un URI :
CCSID

Identificateur (CCSID) du jeu de caractères codés, ou page de codes, dans lequel se trouvent les chaînes de données de type caractères dans le corps d'un message lorsque le client XMS transmet le message à la destination. Lorsqu'elle est définie pour un message individuel, la propriété JMS_IBM_CHARACTER_SET remplace le CCSID spécifié pour la destination par cette propriété.

La valeur par défaut de la propriété est 1208.

Cette propriété est pertinente pour les messages envoyés à la destination, mais pas pour les messages reçus depuis la destination.

XMSC_WMQ_CHANNEL

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: CHANNEL

Nom abrégé de l'outil d'administration JMS: CHAN

Nom du canal à utiliser pour une connexion.

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client.

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: CLIENTRECONNECTOPTIONS

Nom abrégé de l'outil d'administration JMS: CROPT

Cette propriété spécifie les options de reconnexion du client pour les nouvelles connexions créées par cette fabrique. Il se trouve dans XMSC, et est l'un des suivants:

- WMQ_CLIENT_RECONNECT_AS_DEF (par défaut). Utilisez la valeur spécifiée dans le fichier `mqclient.ini`. Définissez la valeur en utilisant la propriété **DefRecon** de la strophe Channels. Il peut s'agir de :
  1. YES. Se comporte comme l'option WMQ_CLIENT_RECONNECT
  2. NO. Valeur par défaut. Ne spécifie aucune option de reconnexion
  3. QMGR. Se comporte comme l'option WMQ_CLIENT_RECONNECT_Q_MGR
  4. DISABLED. Se comporte comme l'option WMQ_CLIENT_RECONNECT_DISABLED
- WMQ_CLIENT_RECONNECT. Se reconnecte à n'importe quel gestionnaire de files d'attente spécifié dans la liste des noms de connexion.
- WMQ_CLIENT_RECONNECT_Q_MGR. Se reconnecte au même gestionnaire de files d'attente avec lequel la connexion d'origine a été établie. `MQRC_RECONNECT_QMID_MISMATCH` est renvoyé si le gestionnaire de files d'attente auquel la tentative de connexion est effectuée (spécifié dans la liste des noms de connexion) a un QMID différent du gestionnaire de files d'attente avec lequel la connexion était établie à l'origine.
- WMQ_CLIENT_RECONNECT_DISABLED. La reconnexion est désactivée.

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: CLIENTRECONNECTTIMEOUT

Nom abrégé de l'outil d'administration JMS: CRT

La propriété XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT est valide uniquement pour le client XMS .NET géré.

Cette propriété spécifie le temps, en secondes, qu'un connexion client utilise pour essayer de se reconnecter.

Passé ce délai, le client échoue avec MQRC_RECONNECT_FAILED. Le paramètre par défaut pour cette propriété est XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT.

La valeur par défaut de cette propriété est 1800.

XMSC_WMQ_CONNECTION_MODE

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Mode par lequel une application se connecte à un gestionnaire de files d'attente.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WMQ_CM_BINDINGS	Connexion à un gestionnaire de files d'attente en mode liaisons, pour une performance optimale. Il s'agit de la valeur par défaut pour C/C++.

Valeur admise	Explication
XMSC_WMQ_CM_CLIENT	Connexion à un gestionnaire de files d'attente en mode client, pour une gestion totale de pile. Il s'agit de la valeur par défaut pour .NET.
XMSC_WMQ_CM_CLIENT_UNMANAGED (pour .NET uniquement)	Connexion à un gestionnaire de files d'attente qui force une pile de client non géré.

Concepts associés

Opérations gérées et non gérées dans .NET

Le code géré est exécuté exclusivement dans la routine d'exécution du langage commun .NET ; il est totalement dépendant des services fournis par cette routine. Une application est classée comme non gérée si une partie de celle-ci exécute ou appelle des services externes à la routine d'exécution du langage commun .NET.

XMSC_WMQ_CONNECTION_NAME_LIST

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: CONNECTIONNAMELIST

Nom abrégé de l'outil d'administration JMS: CNLIST

Cette propriété spécifie les hôtes auxquels le client tente de se reconnecter après une interruption de sa connexion.

La liste de noms de connexion est une liste séparée par des virgules de paires de port hôte/ip. Le paramètre par défaut pour cette propriété est WMQ_CONNECTION_NAME_LIST_DEFAULT.

Par exemple, `127.0.0.1(1414),host2.example.com(1400)`

Le paramètre par défaut de cette propriété est `localhost(1414)`.

XMSC_WMQ_DUR_SUBQ

Type de données :
String

Propriété de :
Destination

Nom de la file d'attente de souscription pour un abonné durable qui reçoit des messages de la destination. Seule une destination représentant une rubrique peut avoir cette propriété.

Le nom de la file d'attente de souscription doit commencer par les caractères suivants :

SYSTEM.JMS.D.

Si vous voulez que tous les abonnés durables partagent une file d'attente de souscription, spécifiez le nom complet de la file d'attente partagée. Une file d'attente du nom spécifié doit exister avant qu'une application puisse créer un abonné durable.

Si vous souhaitez que chaque abonné durable puisse extraire des messages de sa propre file d'attente de souscription exclusive, indiquez un nom de file d'attente qui se termine par un astérisque (*). Ensuite, lorsqu'une application crée un abonné durable, le client XMS crée une file d'attente dynamique à l'usage exclusif de l'abonné durable. Le client XMS utilise la valeur de la propriété pour définir le contenu de la zone **DynamicQName** dans le descripteur d'objet utilisé pour créer la file d'attente dynamique.

La valeur par défaut de la propriété est SYSTEM.JMS.D.SUBSCRIBER.QUEUE, qui signifie que XMS utilise l'approche de file d'attente partagée par défaut.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WMQ_ENCODING

Type de données :
System.Int32

Propriété de :
Destination

Mode de représentation des données numériques dans le corps d'un message lorsque le client XMS transmet le message à la destination. Lorsqu'elle est définie pour un message individuel, la propriété JMS_IBM_ENCODING se substitue au codage spécifié pour la destination. La propriété spécifie la représentation d'entiers binaires, d'entiers décimaux condensés et de nombres à virgule flottante.

Les valeurs valides de la propriété sont les mêmes que celles qui peuvent être spécifiées dans la zone **Encoding** d'un descripteur de message.

Une application peut utiliser les constantes nommées suivantes pour définir la propriété :

Constante nommée	Explication
MQENC_INTEGER_NORMAL	Codage d'entier normal
MQENC_INTEGER_REVERSED	Codage d'entier inversé
MQENC_DECIMAL_NORMAL	Codage décimal condensé normal
MQENC_DECIMAL_REVERSED	Codage décimal condensé inversé
MQENC_FLOAT_IEEE_NORMAL	Codage à virgule flottante IEEE normal
MQENC_FLOAT_IEEE_REVERSED	Codage à virgule flottante IEEE inversé
MQENC_FLOAT_S390	Codage à virgule flottante de l'architecture z/OS
MQENC_NATIVE	Codage machine natif

Pour former une valeur pour la propriété, l'application peut ajouter trois des constantes suivantes :

- Une constante dont le nom commence par MQENC_INTEGER, pour spécifier la représentation d'entiers binaires
- Une constante dont le nom commence par MQENC_DECIMAL, pour spécifier la représentation d'entiers décimaux condensés
- Une constante dont le nom commence par MQENC_FLOAT, pour spécifier la représentation de nombres à virgule flottante

L'application peut également définir la propriété à MQENC_NATIVE, dont la valeur dépend de l'environnement.

La valeur par défaut de la propriété est MQENC_NATIVE.

Cette propriété est pertinente pour les messages envoyés à la destination, mais pas pour les messages reçus depuis la destination.

XMSC_WMQ_FAIL_IF_QUIESCE

Type de données :
System.Int32

Propriété de :
ConnectionFactory et Destination

Nom utilisé dans un URI :
failIfQuiesce

Objets applicables:
Nom long de l'outil d'administration JMS: FAILIFQUIESCE

Nom abrégé de l'outil d'administration JMS: FIQ

Indique que les appels à certaines méthodes échouent si le gestionnaire de files d'attente auquel l'application est connectée est à l'état de mise au repos.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WMQ_FIQ_YES	Les appels à certaines méthodes échouent si le gestionnaire de files d'attente est à l'état de mise au repos. Lorsque l'application détecte que le gestionnaire de files d'attente est au repos, elle termine la tâche immédiate et ferme la connexion, ce qui permet l'arrêt du gestionnaire de files d'attente.
XMSC_WMQ_FIQ_NO	Aucun appel de méthode n'échoue si le gestionnaire de files d'attente est à l'état de mise au repos. Si vous spécifiez cette valeur, l'application ne peut pas détecter que le gestionnaire de files d'attente est à l'état de mise au repos. L'application peut continuer à effectuer des opérations sur le gestionnaire de files d'attente et donc, empêcher l'arrêt de celui-ci.

La valeur par défaut d'une fabrique de connexions est XMSC_WMQ_FIQ_YES mais, aucune destination par défaut n'est définie. La définition de la propriété pour une destination remplace toute valeur spécifiée par la propriété de fabrique de connexions.

XMSC_WMQ_MESSAGE_BODY

Type de données :
System.Int32

Propriété de :
Destination

Cette propriété détermine si une application XMS traite l' MQRFH2 d'un message IBM WebSphere MQ dans la charge du message (c'est-à-dire dans le corps du message).

Remarque : Lors de l'envoi de messages vers une destination, la propriété XMSC_WMQ_MESSAGE_BODY remplace la propriété de destination XMS existante XMSC_WMQ_TARGET_CLIENT.

Les valeurs valides de cette propriété sont les suivantes :

XMSC_WMQ_MESSAGE_BODY_JMS
Réception : Le type et le corps du message XMS entrant sont déterminés par le contenu du MQRFH2 (le cas échéant) ou du MQMD (en l'absence de MQRFH2) dans le message IBM WebSphere MQ reçu.

Envoi : Le corps du message XMS sortant contient un en-tête MQRFH2 ajouté en préfixe et généré automatiquement basé sur les propriétés et les en-têtes de message XMS.

XMSC_WMQ_MESSAGE_BODY_MQ
Réception : Le type de message XMS entrant est toujours ByteMessage, indépendamment du contenu du message IBM WebSphere MQ reçu ou de la zone de format du MQMD reçu. Le corps du message XMS est constitué des données de message non modifiées renvoyées par l'appel API du fournisseur de messagerie sous-jacent. Le jeu de caractères et l'encodage des données dans le corps du message sont déterminés par les zones CodedCharSetId et Encoding fields du MQMD. Le format des données dans le corps du message est déterminé par la zone Format field du MQMD.

Envoi : Le corps du message XMS sortant contient le contenu de l'application tel quel ; aucun en-tête IBM WebSphere MQ généré automatiquement n'est ajouté au corps.

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED
Réception : Le client XMS détermine une valeur adaptée pour cette propriété. Sur un chemin de réception, cette valeur est celle de la propriété WMQ_MESSAGE_BODY_JMS.

Envoi : Le client XMS détermine une valeur adaptée pour cette propriété. Sur un chemin d'envoi, cette valeur est celle de la propriété XMSC_WMQ_TARGET_CLIENT.

Par défaut, cette propriété est définie sur XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED.

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

Type de données :
System.Int32

Propriété de :
Destination

Détermine le niveau de contexte de message à définir par l'application XMS . L'application doit être exécutée avec les droits appropriés pour que cette propriété soit appliquée.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ_MDCTX_DEFAULT
Pour les messages sortants, l'appel API MQOPEN API et la structure MQPMO ne spécifient pas d'options de contexte de message explicites.

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT
L'appel API MQOPEN API spécifie l'option de contexte de message MQOO_SET_IDENTITY_CONTEXT et la structure MQPMO spécifie MQPMO_SET_IDENTITY_CONTEXT.

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT
L'appel API MQOPEN API spécifie l'option de contexte de message MQOO_SET_ALL_CONTEXT et la structure MQPMO spécifie MQPMO_SET_ALL_CONTEXT.

Par défaut, cette propriété est définie sur XMSC_WMQ_MDCTX_DEFAULT.

Remarque : Cette propriété n'est pas pertinente lorsqu'une application se connecte au bus d'intégration de services.

Les propriétés suivantes nécessitent que la propriété XMSC_WMQ_MQMD_MESSAGE_CONTEXT soit définie sur la valeurXMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT ou XMSC_WMQ_MDCTX_SET_ALL_CONTEXT lors de l'envoi d'un message :

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN
- JMS_IBM_MQMD_APPLIDENTITYDATA

Les propriétés suivantes nécessitent que la propriété XMSC_WMQ_MQMD_MESSAGE_CONTEXT soit définie sur XMSC_WMQ_MDCTX_SET_ALL_CONTEXT lors de l'envoi d'un message :

- JMS_IBM_MQMD_PUTAPPLTYPE
- JMS_IBM_MQMD_PUTAPPLNAME
- JMS_IBM_MQMD_PUTDATE
- JMS_IBM_MQMD_PUTTIME
- JMS_IBM_MQMD_APPLORIGINDATA

XMSC_WMQ_MQMD_READ_ENABLED

Type de données :
System.Int32

Propriété de :
Destination

Cette propriété détermine si une application XMS peut ou non extraire les valeurs des zones MQMD.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ_READ_ENABLED_NO
Lors de l'envoi de messages, les propriétés JMS_IBM_MQMD* d'un message envoyé ne sont pas mises à jour pour refléter les valeurs de zone mises à jour dans le MQMD.

Lors de la réception de messages, aucune propriété JMS_IBM_MQMD* d'un message reçu n'est disponible, même si elles sont en partie ou toutes définies par l'émetteur.

XMSC_WMQ_READ_ENABLED_YES
Lors de l'envoi de messages, toutes les propriétés JMS_IBM_MQMD* d'un message envoyé sont mises à jour pour refléter les valeurs de zone mises à jour dans le MQMD, y compris les propriétés que l'émetteur n'a pas explicitement définies.

Lors de la réception de messages, toutes les propriétés JMS_IBM_MQMD* d'un message reçu sont disponibles, y compris les propriétés que l'émetteur n'a pas explicitement définies.

Par défaut, cette propriété est définie sur XMSC_WMQ_READ_ENABLED_NO.

XMSC_WMQ_MQMD_WRITE_ENABLED

Type de données :
System.Int32

Propriété de :
Destination

Cette propriété détermine si une application XMS peut extraire les valeurs des zones MQMD.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ_WRITE_ENABLED_NO
Toutes les propriétés JMS_IBM_MQMD* sont ignorées et leurs valeurs ne sont pas copiées dans la structure MQMD sous-jacente.

XMSC_WMQ_WRITE_ENABLED_YES
Les propriétés JMS_IBM_MQMD* sont traitées. Leurs valeurs sont copiées dans la structure sous-jacente.

Par défaut, cette propriété est définie sur XMSC_WMQ_WRITE_ENABLED_NO.

XMSC_WMQ_PUT_ASYNC_ALLOWED

Type de données :
System.Int32

Propriété de :
Destination

Cette propriété détermine si les expéditeurs de message sont autorisés à utiliser les insertions asynchrones pour envoyer des messages à cette destination.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST
Détermine si les insertions asynchrones sont autorisées conformément à la définition de rubrique ou de file d'attente.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_Q_DEF
Détermine si les insertions asynchrones sont autorisées conformément à la définition de file d'attente.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_TOPIC_DEF
Détermine si les insertions asynchrones sont autorisées conformément à la définition de rubrique.

XMSC_WMQ _PUT_ASYNC_ALLOWED_DISABLED
Les insertions asynchrones ne sont pas autorisées.

XMSC_WMQ _PUT_ASYNC_ALLOWED_ENABLED
Les insertions asynchrones sont autorisées.

Par défaut, cette propriété est définie sur XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_DEST.

Remarque : Cette propriété n'est pas appropriée lorsqu'une application se connecte au bus d'intégration de services.

XMSC_WMQ_READ_AHEAD_ALLOWED

Type de données :
System.Int32

Propriété de :
Destination

Cette propriété détermine si les consommateurs de message et les navigateurs de files d'attente sont autorisés à utiliser la lecture anticipée pour extraire les messages non permanents et non transactionnels de cette destination dans une mémoire tampon interne avant de les recevoir.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF
Détermine si la lecture anticipée est autorisée conformément à la définition de file d'attente.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_ TOPIC _DEF
Détermine si la lecture anticipée est autorisée conformément à la définition de rubrique.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST
Détermine si la lecture anticipée est autorisée conformément à la définition de file d'attente ou de rubrique.

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED
La lecture anticipée n'est pas autorisé lors de la consommation ou de la consultation de messages.

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED
La lecture anticipée est autorisée.

Par défaut, cette propriété est définie sur XMSC_WMQ _READ_AHEAD_ALLOWED_AS_DEST.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

Type de données :
System.Int32

Propriété de :
Destination

Cette propriété détermine, pour les messages distribués à un programme d'écoute de messages asynchrone, ce qu'il advient des messages dans la mémoire tampon interne de lecture anticipée lorsque le consommateur de messages est fermé.

Cette propriété est applicable en spécifiant les options de fermeture de file d'attente lors de la consommation des messages en provenance d'une destination, mais non applicable lors de l'envoi de messages vers une destination.

Cette propriété est ignorée pour les navigateurs de files d'attente, les messages étant toujours disponibles dans la file d'attente pendant l'exploration.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT
Seul l'appel du programme d'écoute des messages se termine avant d'être renvoyé, laissant potentiellement des messages dans la mémoire tampon interne de lecture anticipée qui sont ensuite supprimés.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL
Tous les messages dans la mémoire tampon interne de lecture anticipée sont distribués au programme d'écoute des messages de l'application avant d'être renvoyés.

Par défaut, cette propriété est définie sur XMSC_WMQ
_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT.

Remarque :

- **Arrêt anomal de l'application**

  Tous les messages dans la mémoire tampon de lecture anticipée sont perdus lorsqu'une application XMS s'arrête soudainement.

- **Implications sur les transactions**

  La lecture anticipée est désactivée lorsque les applications utilisent la transaction. L'application ne voit donc aucune différence dans le comportement lorsqu'elles utilisent les sessions transactionnelles.

- **Implications des modes d'accusé de réception de session**

  La lecture anticipée est activée sur une session non transactionnelle lorsque le mode d'accusé de réception est XMSC_AUTO_ACKNOWLEDGE ou XMSC_DUPS_OK_ACKNOWLEDGE. La lecture anticipée est désactivée si le mode d'accusé de réception de la session est XMSC_CLIENT_ACKNOWLEDGE, que la session soit transactionnelle ou non.

- **Implications sur les navigateurs de files d'attente et les sélecteurs de navigateur de files d'attente**

  Les navigateurs de files d'attente et les sélecteurs de navigateur de files d'attente utilisés dans les applications XMS tirent profit de la lecture anticipée. La fermeture du navigateur de files d'attente n'affecte pas les performances, car le message reste disponible en file d'attente pour de futures opérations. En dehors des avantages de performances liés à la lecture anticipée, il n'existe pas d'autre implication pour les navigateurs de files d'attente et les sélecteurs de navigateur de files d'attente.

XMSC_WMQ_HOST_NAME

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: HOSTNAME

Nom abrégé de l'outil d'administration JMS: HOST

Nom d'hôte ou adresse IP du système sur lequel s'exécute un gestionnaire de files d'attente.

Cette propriété est utilisée uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client. La propriété est utilisée avec la propriété XMSC_WMQ_PORT pour identifier le gestionnaire de files d'attente.

La valeur par défaut de la propriété est `localhost`.

XMSC_WMQ_LOCAL_ADDRESS

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: LOCALADDRESS

Nom abrégé de l'outil d'administration JMS: LA

Pour une connexion à un gestionnaire de files d'attente, cette propriété spécifie l'interface de réseau local et/ou le port ou la plage de ports à utiliser.

La valeur de la propriété est une chaîne au format suivant :

[*nom_hôte*][(*port_bas*)[,*port_haut*])]

La signification des variables est la suivante :

nom_hôte
Nom d'hôte ou adresse IP de l'interface de réseau local à utiliser pour la connexion.

Cette information est requise uniquement si le système sur lequel l'application s'exécute dispose d'au moins deux interfaces réseau et que vous devez pouvoir spécifier l'interface qui doit être utilisée pour la connexion. Si le système dispose d'une seule interface réseau, seule cette interface peut être utilisée. Si le système dispose d'au moins deux interfaces réseau et que vous ne spécifiez pas quelle interface doit être utilisée, celle-ci est sélectionnée au hasard.

port_bas
Numéro du port local à utiliser pour la connexion.

Si *port_haut* est aussi spécifiée, *port_bas* est interprétée comme correspondant au numéro de port le plus bas de la plage de ports.

port_haut
Numéro de port le plus élevé dans la plage de ports. Un des ports de la plage spécifiée doit être utilisé pour la connexion.

La longueur maximale de la chaîne est de 48 caractères.

Voici quelques exemples de valeurs valides pour la propriété :

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client.

XMSC_WMQ_MESSAGE_SELECTION

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Détermine si la sélection des messages est effectuée par le client XMS ou par le courtier.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WMQ_MSEL_CLIENT	La sélection de message est faite par le client XMS.
XMSC_WMQ_MSEL_BROKER	La sélection de message est faite par le courtier.

La valeur par défaut est XMSC_WMQ_MSEL_CLIENT.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner. La sélection de message par le courtier n'est pas prise en charge si la propriété XMSC_WMQ_BROKER_VERSION est définie sur XMSC_WMQ_BROKER_V1.

XMSC_WMQ_MSG_BATCH_SIZE

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Nombre maximal de messages à extraire d'une file d'attente en un seul lot lorsque la distribution asynchrone de messages est utilisée.

Lorsqu'une application utilise la distribution asynchrone de messages, sous certaines conditions, le client XMS extrait un lot de messages d'une file d'attente avant de réacheminer chaque message de manière individuelle à l'application. Cette propriété spécifie le nombre maximal de messages pouvant constituer un lot.

Il s'agit d'un entier positif ; la valeur par défaut est 10. Envisagez de définir une autre valeur pour cette propriété uniquement si vous devez résoudre un problème spécifique de performance.

Si une application est connectée à un gestionnaire de files d'attente via un réseau, l'augmentation de la valeur de cette propriété peut réduire les surcharges réseau et les temps de réponse, mais augmenter la quantité de mémoire requise pour stocker les messages sur un système client. A l'inverse, la diminution de la valeur de cette propriété peut augmenter les surcharges réseau et les temps de réponse, mais réduire la quantité de mémoire requise pour stocker les messages.

XMSC_WMQ_POLLING_INTERVAL

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Si chaque programme d'écoute de message dans une session ne comporte pas de message approprié dans sa file d'attente, cette valeur est l'intervalle de temps maximal, en millisecondes, qui s'écoule avant que chaque programme d'écoute de message essaie à nouveau d'obtenir un message à partir de sa file d'attente.

Si l'absence de message approprié est fréquemment observée pour l'un quelconque des écouteurs de messages au sein d'une session, envisagez d'augmenter la valeur de cette propriété.

La valeur de la propriété est un entier positif. La valeur par défaut est 5 000.

XMSC_WMQ_PORT

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: PORT

Nom abrégé de l'outil d'administration JMS: PORT

Numéro du port sur lequel un gestionnaire de files d'attente écoute les demandes entrantes.

Cette propriété est utilisée uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client. La propriété est utilisée avec la propriété XMSC_WMQ_HOST_NAME pour identifier le gestionnaire de files d'attente.

La valeur par défaut de la propriété est XMSC_WMQ_DEFAULT_CLIENT_PORT, ou 1414.

XMSC_WMQ_PROVIDER_VERSION

Type de données :
String

Propriété de :
ConnectionFactory

Version, édition, niveau de modification et groupe de correctifs du gestionnaire de files d'attente auquel l'application a l'intention de se connecter. Les valeurs valides de cette propriété sont les suivantes :

- Non spécifié

Ou une chaîne dans l'un des formats suivants

- V.R.M.F
- V.R.M
- V.R
- V

où V, R, M et F sont des entiers strictement positifs.

Une valeur supérieure ou égale à 7 indique que cette version est conçue comme une IBM WebSphere MQ 7.0 ConnectionFactory pour les connexions à un gestionnaire de files d'attente IBM WebSphere MQ 7.0 . Une valeur inférieure à 7 (par exemple, "6.0.2.0") indique que le client sera utilisé avec les gestionnaires de files d'attente antérieurs à la version 7.0. La valeur par défaut (Non spécifié) signifie que toutes les connexions peuvent être établies vers un gestionnaire de files d'attentes, sans tenir compte de sa version, et que les propriétés et les fonctionnalités disponibles sont fonction des capacités du gestionnaire de files d'attente utilisé.

Par défaut, cette propriété n'est pas spécifiée.

Remarque :

- Aucun partage de socket n'a lieu si XMSC_WMQ_PROVIDER_VERSION est défini sur 6.2.
- La connexion échoue si XMSC_WMQ_PROVIDER_VERSION est défini sur 7 et sur le serveur SHARECNV pour le canal est défini sur 0.
- Les fonctions IBM WebSphere MQ 7.0 spécifiques sont désactivées si XMSC_WMQ_PROVIDER_VERSION est défini surUNSPECIFIED et SHARECNV est défini sur 0.

La version du client IBM WebSphere MQ joue également un rôle majeur dans la possibilité pour une application XMS client d'utiliser les fonctions IBM WebSphere MQ 7.0 spécifiques. Le tableau suivant décrit ce comportement.

Remarque : Une propriété système XMSC_WMQ_OVERRIDEPROVIDERVERSION se substitue à la propriété XMSC_WMQ_PROVIDER_VERSION. Cette propriété peut être utilisée si vous ne pouvez pas modifier le paramètre de fabrique de connexions.

Tableau 33. Client XMS - Possibilité d'utilisation des fonctions IBM WebSphere MQ 7.0 spécifiques.

#	XMSC_WMQ_PROVIDER_VERSION	Version du client IBM WebSphere MQ	Fonctions de IBM WebSphere MQ 7.0
1	non spécifié	7	Oui
2	non spécifié	6	Non
3	7	7	Oui
4	7	6	Exception
5	6	6	Non
6	6	7	Non

XMSC_WMQ_PUB_ACK_INTERVAL

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Nombre de messages publiés par un diffuseur de publications avant que le client XMS ne demande un accusé de réception du courtier.

Si vous réduisez la valeur de cette propriété, le client demande des accusés de réception plus souvent et, par conséquent, les performances du diffuseur de publications diminuent. Si vous augmentez cette valeur, le client observe un délai plus long avant d'émettre une exception sur un échec du courtier.

La valeur de la propriété est un entier positif. La valeur par défaut est 25.

XMSC_WMQ_QMGR_CCSID

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Identificateur (CCSID) du jeu de caractères codés, ou page de codes, dans lequel les zones de données de type caractères définies dans l'interface MQI (Message Queue Interface) sont échangées entre le client XMS et le client WebSphere MQ . Cette propriété ne s'applique pas pour les chaînes de données de type caractères du corps des messages.

Lorsque l'application Un XMS se connecte à un gestionnaire de files d'attente en mode client, le client XMS se connecte au client WebSphere MQ . Les informations échangées entre les deux clients contiennent des zones de données de type caractères définies dans l'interface MQI. Dans des circonstances normales, le client WebSphere MQ considère que ces zones sont dans la page de codes du système sur lequel les clients sont exécutés. Si le client XMS envoie et attend de recevoir ces zones dans une page de codes différente, vous devez définir cette propriété pour en informer le client WebSphere MQ.

Lorsque le client WebSphere MQ réachemine ces zones de données de type caractères vers le gestionnaire de files d'attente, les données doivent être converties si nécessaire dans la page de codes utilisée par le gestionnaire de files d'attente. De même, lorsque le client WebSphere MQ reçoit ces zones en provenance du gestionnaire de files d'attente, les données doivent être converties si nécessaire dans la page de codes dans laquelle le client XMS s'attend à les recevoir. Le client WebSphere MQ utilise cette propriété pour effectuer les conversions de données.

Par défaut, la propriété n'est pas définie.

Définir cette propriété équivaut à définir la variable d'environnement MQCCSID pour un client WebSphere MQ qui prend en charge des applications client WebSphere MQ natives. Pour plus d'informations sur cette variable d'environnement, voir *Clients WebSphere MQ*.

XMSC_WMQ_QUEUE_MANAGER

Type de données :
String

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: QMANAGER

Nom abrégé de l'outil d'administration JMS: QMGR

Nom du gestionnaire de files d'attente auquel se connecter.

Par défaut, la propriété n'est pas définie.

XMSC_WMQ_RECEIVE_CCSID

Propriété de destination qui définit le CCSID cible pour la conversion des messages du gestionnaire de files d'attente. La valeur est ignorée sauf si XMSC_WMQ_RECEIVE_CONVERSION est défini sur WMQ_RECEIVE_CONVERSION_QMGR.

Type de données :
Entier

Valeur :
Tout nombre entier positif.

La valeur par défaut est 1208.

La spécification d'une valeur GMO_CONVERT dans un message est facultative. Si une valeur GMO_CONVERT est spécifiée, la conversion s'effectue en fonction de cette valeur.

XMSC_WMQ_RECEIVE_CONVERSION

Propriété de destination qui détermine si la conversion de données sera effectuée par le gestionnaire de files d'attente.

Type de données :
Entier

Valeurs :
XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (valeur par défaut) : effectue les conversions de données uniquement sur le client XMS. La conversion est toujours effectuée avec la page de codes 1208.

XMSC_WMQ_RECEIVE_CONVERSION_QMGR : effectue les conversions de données sur le gestionnaire de files d'attente avant d'envoyer un message au client XMS.

XMSC_WMQ_RECEIVE_EXIT

Type de données :
String

Propriété de :
ConnectionFactory

Identifie un exit de réception de canal à exécuter.

La valeur de la propriété est une chaîne qui identifie un exit de réception de canal et a le format suivant :

libraryName(entryPointName)

où

- **libraryName** correspond au chemin d'accès complet du fichier .dll d'exit géré
- entryPointName correspond au nom de classe qualifié par l'espace de nom

Par exemple, C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client géré. En outre, seuls les exits gérés sont pris en charge.

XMSC_WMQ_RECEIVE_EXIT_INIT

Type de données :
String

Propriété de :
ConnectionFactory

Données utilisateur transmises à un exit de réception de canal lors de l'appel.

La valeur de la propriété est une chaîne. Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client géré et que la propriété «XMSC_WMQ_RECEIVE_EXIT», à la page 232 est définie.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

Type de données :
String

Propriété de :
ConnectionFactory

Cette propriété est utilisée pour obtenir le nom du gestionnaire de files d'attente auquel il est connecté.

Lorsqu'il est utilisé avec une table de définition de canal du client, ce nom peut être différent de celui du gestionnaire de files d'attente de la fabrique de connexions.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

Type de données :
String

Propriété de :
ConnectionFactory

Cette propriété est remplie avec l'ID du gestionnaire de files d'attente après la connexion.

XMSC_WMQ_SECURITY_EXIT

Type de données :
String

Propriété de :
ConnectionFactory

Identifie un exit de sécurité de canal.

La valeur de la propriété est une chaîne qui identifie un exit de sécurité de canal et a le format suivant :

libraryName(entryPointName)

où

- **libraryName** correspond au chemin d'accès complet du fichier .dll d'exit géré
- entryPointName correspond au nom de classe qualifié par l'espace de nom

Par exemple, C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

La longueur maximale de la chaîne est de 128 caractères.

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client géré. En outre, seuls les exits gérés sont pris en charge.

XMSC_WMQ_SECURITY_EXIT_INIT

Type de données :
String

Propriété de :
ConnectionFactory

Données utilisateur transmises à un exit de sécurité de canal lors de l'appel.

La longueur maximale de la chaîne de données utilisateur est de 32 caractères.

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client géré et que la propriété «XMSC_WMQ_SECURITY_EXIT», à la page 233 est définie.

XMSC_WMQ_SEND_EXIT

Type de données :
String

Propriété de :
ConnectionFactory

Identifie un exit d'émission de canal.

La valeur de la propriété est une chaîne. Un exit d'émission de canal a le format suivant :

libraryName(entryPointName)

où

- **libraryName** correspond au chemin d'accès complet du fichier .dll d'exit géré
- entryPointName correspond au nom de classe qualifié par l'espace de nom

Par exemple, C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client géré. En outre, seuls les exits gérés sont pris en charge.

XMSC_WMQ_SEND_EXIT_INIT

Type de données :
String

Propriété de :
ConnectionFactory

Données utilisateur qui sont transmises aux exits d'émission une fois ceux-ci appelés.

La valeur de la propriété est une chaîne de un ou plusieurs éléments de données utilisateur séparés par des virgules. Par défaut, la propriété n'est pas définie.

Les règles de spécification des données utilisateur qui sont transmises à une séquence d'exits d'émission de canal sont les mêmes que les règles de spécification de données utilisateur transmises à une séquence d'exits de réception de canal. Pour les règles, voir «XMSC_WMQ_RECEIVE_EXIT_INIT», à la page 232.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client géré et que la propriété «XMSC_WMQ_SEND_EXIT», à la page 234 est définie.

XMSC_WMQ_SEND_CHECK_COUNT

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Nombre d'appels d'envoi à autoriser entre deux vérifications d'erreurs asynchrones, au cours d'une même session XMS non transactionnelle.

Par défaut, cette propriété est définie sur 0.

XMSC_WMQ_SHARE_CONV_ALLOWED

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Objets applicables:
Nom long de l'outil d'administration JMS: SHARECONVALLOWED

Nom abrégé de l'outil d'administration JMS: SCALD

Indique si une connexion client peut partager son socket avec d'autres connexions XMS de niveau supérieur à partir du même processus vers le même gestionnaire de files d'attente, si les définitions de canal correspondent. Cette propriété est fournie pour permettre un isolement complet des connexions dans des sockets distincts, lorsque cela est requis pour des raisons de développement, de maintenance ou de fonctionnement. La définition de cette propriété indique à XMS de partager le socket sous-jacent. Elle n'indique pas le nombre de connexions partageant un socket. Ceci est déterminé par la valeur SHARECNV, négociée entre WebSphere MQ Client et WebSphere MQ Server.

Une application peut utiliser les constantes nommées suivantes pour définir la propriété :

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - Les connexions ne partagent pas de socket.
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - Les connexions partagent un socket.

Par défaut, cette propriété est définie sur XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED.

Cette propriété est pertinente uniquement lorsqu'une application se connecte à un gestionnaire de files d'attente en mode client.

XMSC_WMQ_SSL_CERT_STORES

Type de données :
String

Propriété de :
ConnectionFactory

Emplacements des serveurs contenant les listes de révocation de certificat à utiliser sur une connexion SSL à un gestionnaire de files d'attente.

La valeur de la propriété est une liste de une ou plusieurs adresses URL séparées par des virgules. Chaque adresse URL a le format suivant :

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

Ce format est compatible avec le format MQJMS de base.

La valeur `serveraddress`doit être vide. Dans ce cas, XMS utilise la chaîne "localhost".

Exemple :

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

Par défaut, la propriété n'est pas définie.

Information associée
SSL and TLS support for the unmanaged .NET client
SSL and TLS support for the managed .NET client

XMSC_WMQ_SSL_CIPHER_SPEC

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la spécification CipherSpec à utiliser sur une connexion sécurisée vers un gestionnaire de files d'attente.

Les spécifications de chiffrement que vous pouvez utiliser avec le support TLS d'IBM WebSphere MQ sont répertoriées dans le tableau ci-dessous. Lorsque vous demandez un certificat personnel, vous définissez une taille de clé pour la paire de clé publique et de clé privée. La taille de clé utilisée lors de l'établissement de liaison SSL est celle qui est stockée dans le certificat à moins qu'elle soit déterminée par la spécification CipherSpec, comme indiqué dans le tableau. Par défaut, cette propriété n'est pas définie.

Nom du CipherSpec	Protocole utilisé	Algorithme de hachage	Algorithme de chiffrement	Bits de chiffrement	FIPS [1]	Suite B 128 bits	Suite B 192 bits
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	Norme AES	128	Oui	Non	Non
TLS_RSA_WITH_AES_256_CBC_SHA[2]	TLS 1.0	SHA-1	Norme AES	256	Oui	Non	Non
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	Non	Non	Non
TLS_RSA_WITH_3DES_EDE_CBC_SHA[4]	TLS 1.0	SHA-1	3DES	168	Oui	Non	Non
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	Norme AES	128	Oui	Non	Non
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	Norme AES	256	Oui	Non	Non
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	Norme AES	128	Oui	Non	Non
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	Norme AES	256	Oui	Non	Non
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Non	Non	Non
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Oui	Non	Non
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Non	Non	Non
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Oui	Non	Non
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	Norme AES	128	Oui	Non	Non
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	Norme AES	256	Oui	Non	Non
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	Norme AES	128	Oui	Non	Non
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	Norme AES	256	Oui	Non	Non
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	Norme AES	128	Oui	Oui	Non

Nom du CipherSpec	Protocole utilisé	Algorithme de hachage	Algorithme de chiffrement	Bits de chiffrement	FIPS [1]	Suite B 128 bits	Suite B 192 bits
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	Norme AES	256	Oui	Non	Oui
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	Norme AES	128	Oui	Non	Non
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	Norme AES	256	Oui	Non	Non
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	Aucun	0	Non	Non	Non
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	Aucun	0	Non	Non	Non
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	Aucun	0	Non	Non	Non
TLS_RSA_WITH_NULL_NULL	TLS 1.2	Aucun	Aucun	0	Non	Non	Non
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Non	Non	Non

Remarques :

1. Indique si la spécification CipherSpec est conforme à la norme FIPS (Federal Information Processing Standards) 140-2. Pour plus d'informations sur la norme FIPS et sur la configuration de WebSphere MQ pour une opération conforme à FIPS 140-2, reportez-vous à *Federal Information Processing Standards (FIPS)* dans la documentation en ligne du produit IBM IBM WebSphere MQ.
2. Cette spécification CipherSpec ne peut pas être utilisée pour sécuriser une connexion de WebSphere MQ Explorer à un gestionnaire de files d'attente, sauf si les fichiers de règles sans restriction appropriés sont appliqués à l'environnement d'exécution Java utilisé par l'explorateur.
3. Ce CipherSpec a été certifié FIPS 140-2 avant le 19 mai 2007.
4. Lorsque WebSphere MQ est configuré pour une opération conforme à FIPS 140-2, cette spécification CipherSpec peut être utilisée pour transférer jusqu'à 32 Go de données avant que la connexion ne se termine avec une erreur AMQ9288. Pour éviter cette erreur, n'utilisez pas la norme DES triple (qui est dépréciée) ou activez la réinitialisation de clé confidentielle lors de l'utilisation de ce CipherSpec dans une configuration FIPS 140-2.

Information associée

Sécurisation

Intégrité des données de messages

Définition des spécifications CipherSpec

XMSC_WMQ_SSL_CIPHER_SUITE

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la suite de chiffrement à utiliser sur une connexion TLS à un gestionnaire de files d'attente. Le protocole utilisé lors de la négociation de la connexion sécurisée dépend de la suite de chiffrement spécifiée.

Cette propriété a les valeurs canoniques suivantes :

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

Cette valeur peut être fournie comme alternative à XMSC_WMQ_SSL_CIPHER_SPEC.

Si une valeur non vide est spécifiée pour XMSC_WMQ_SSL_CIPHER_SPEC, elle remplace le paramétrage pour XMSC_WMQ_SSL_CIPHER_SUITE. Si XMSC_WMQ_SSL_CIPHER_SPEC n'a pas de valeur, la valeur de XMSC_WMQ_SSL_CIPHER_SUITE est utilisée comme algorithme de cryptographie pour GSKit. Dans ce cas, la valeur est mappée en son équivalent CipherSpec, comme décrit dans «Mappages des noms de suite de chiffrement et de CipherSpec pour les connexions à un gestionnaire de files d'attente IBM MQ», à la page 68.

Si XMSC_WMQ_SSL_CIPHER_SPEC et XMSC_WMQ_SSL_CIPHER_SUITE sont vides, la zone pChDef->SSLCipherSpec est complétée avec des espaces.

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

Par défaut, la propriété n'est pas définie.

Information associée

SSL and TLS support for the unmanaged .NET client

SSL and TLS support for the managed .NET client

XMSC_WMQ_SSL_CRYPTO_HW

Type de données :
String

Propriété de :
ConnectionFactory

Détails de configuration pour le matériel de cryptographie connecté au système client.

Cette propriété a les valeurs canoniques suivantes :

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON

Il existe un format spécial pour le matériel de cryptographie PKCS11 (où DriverPath, TokenLabel et TokenPassword sont des chaînes spécifiées par l'utilisateur) :

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS n'interprète ni ne modifie le contenu de la chaîne. Il copie la valeur fournie, dans la limite de 256 caractères à un octet, dans la zone MQSCO.CryptoHardware.

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

Par défaut, la propriété n'est pas définie.

Information associée

SSL and TLS support for the unmanaged .NET client
SSL and TLS support for the managed .NET client

XMSC_WMQ_SSL_FIPS_REQUIRED

Type de données :
Boolean

Propriété de :
ConnectionFactory

La valeur de cette propriété détermine si une application peut ou non utiliser des suites de chiffrement conformes non FIPS. Si cette propriété est pour valeur true, seuls les algorithmes FIPS sont utilisés pour la connexion client/serveur.

Cette propriété peut avoir les valeurs suivantes, qui correspondent aux deux valeurs canoniques pour MQSCO.FipsRequired :

Tableau 34. Tableau de valeurs pour la propriété MQSCO.FlipsRequired

Valeur	Description	Valeur correspondante de MQSCO.FipsRequired
false	Tout CipherSpec peut être utilisé.	MQSSL_FIPS_NO (valeur par défaut)
conforme	Seuls les algorithmes de cryptographie certifiés FIPS peuvent être utilisés dans le CipherSpec s'appliquant à cette connexion client.	MQSSL_FIPS_YES

XMS copie la valeur appropriée dans MQSCO.FipsRequired avant d'appeler MQCONNX.

Le paramètre MQSCO.FipsRequired est disponible uniquement à partir de WebSphere MQ version 6. Si vous définissez cette propriété dans WebSphere MQ version 5.3, XMS n'essaie pas d'établir la connexion avec le gestionnaire de files d'attente, mais envoie une exception appropriée.

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

Information associée

SSL and TLS support for the unmanaged .NET client
SSL and TLS support for the managed .NET client

XMSC_WMQ_SSL_KEY_REPOSITORY

Type de données :
String

Propriété de :
ConnectionFactory

Emplacement du fichier de clés dans lequel les clés et les certificats sont stockés.

XMS copie la chaîne, jusqu'à une limite de 256 caractères à un octet, dans la zone MQSCO.KeyRepository. WebSphere MQ interprète cette chaîne comme un nom de fichier, chemin d'accès complet compris.

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

Par défaut, la propriété n'est pas définie.

Information associée

SSL and TLS support for the unmanaged .NET client

SSL and TLS support for the managed .NET client

XMSC_WMQ_SSL_KEY_RESETCOUNT

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Nombre total d'octets non chiffrés envoyés et reçus dans une conversation SSL avant renégociation de la clé confidentielle. Le nombre d'octets inclut les informations de contrôle envoyées par l'agent MCA.

XMS copie la valeur que vous indiquez pour cette propriété dans MQSCO.KeyResetCount avant d'appeler MQCONNX.

Le paramètre MQSCO.KeyRestCount est disponible uniquement à partir de WebSphere MQ version 6. Si vous définissez cette propriété dans WebSphere MQ version 5.3, XMS n'essaie pas d'établir la connexion avec le gestionnaire de files d'attente, mais envoie une exception appropriée.

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

La valeur par défaut de cette propriété est 0, ce qui signifie que les clés confidentielles ne sont jamais renégociées.

Information associée

SSL and TLS support for the unmanaged .NET client

SSL and TLS support for the managed .NET client

XMSC_WMQ_SSL_PEER_NAME

Type de données :
String

Propriété de :
ConnectionFactory

Nom d'homologue à utiliser sur une connexion SSL vers un gestionnaire de files d'attente.

Il n'existe pas de liste de valeurs canoniques pour cette propriété. A la place, vous devez générer cette chaîne conformément aux règles pour SSLPEER.

Exemple de nom d'homologue :

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS copie la chaîne dans la page de codes à un octet requise, puis positionne les valeurs dans MQCD.SSLPeerNamePtr et MQCD.SSLPeerNameLength avant d'appeler MQCONNX.

Cette propriété est pertinente uniquement si l'application se connecte à un gestionnaire de files d'attente en mode client.

Pour .NET uniquement: depuis IBM MQ 8.0, les connexions gérées à IBM MQ (WMQ_CM_CLIENT) et les connexions non gérées à IBM MQ (WMQ_CM_CLIENT_UNMANAGED) prennent en charge les connexions TLS/SSL.

Par défaut, la propriété n'est pas définie.

Information associée

SSL and TLS support for the unmanaged .NET client
SSL and TLS support for the managed .NET client
SSLPEERNAME

XMSC_WMQ_SYNCPOINT_ALL_GETS

Type de données :
System.Boolean

Propriété de :
ConnectionFactory

Indique si tous les messages doivent être extraits des files d'attente sous le contrôle d'un point de synchronisation.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
false	Lorsque les circonstances le permettent, le client XMS peut extraire les messages des files d'attente sans le contrôle d'un point de synchronisation.
conforme	Le client XMS doit extraire tous les messages des files d'attente sous le contrôle d'un point de synchronisation.

La valeur par défaut est false.

XMSC_WMQ_TARGET_CLIENT

Type de données :
System.Int32

Propriété de :
Destination

Nom utilisé dans un URI :
targetClient

Indique si les messages envoyés à la destination contiennent un en-tête MQRFH2.

Si une application envoie un message contenant un en-tête MQRFH2, l'application destinataire doit être capable de gérer cet en-tête.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WMQ_TARGET_DEST_JMS	Les messages envoyés à la destination contiennent un en-tête MQRFH2. Spécifiez cette valeur si l'application envoie les messages à une autre application XMS, à une application WebSphere JMS ou à une application WebSphere MQ native conçue pour gérer les en-têtes MQRFH2.
XMSC_WMQ_TARGET_DEST_MQ	Les messages envoyés à la destination ne contiennent pas d'en-tête MQRFH2. Spécifiez cette valeur si l'application envoie les messages à une application WebSphere MQ native qui n'est pas conçue pour gérer les en-têtes MQRFH2.

La valeur par défaut est XMSC_WMQ_TARGET_DEST_JMS.

XMSC_WMQ_TEMP_Q_PREFIX

Type de données :
String

Propriété de :
ConnectionFactory

Préfixe utilisé pour former le nom de la file d'attente dynamique WebSphere MQ créée lorsque l'application crée une file d'attente temporaire Un XMS .

Les règles de formation du préfixe sont les mêmes que les règles de formation du contenu de la zone **DynamicQName** dans un descripteur d'objet, mais le dernier caractère non blanc doit être un astérisque (*). Si la propriété n'est pas définie, la valeur utilisée est `CSQ.*` sur z/OS et `AMQ.*` sur les autres plateformes. Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement dans le domaine point-à-point.

XMSC_WMQ_TEMP_TOPIC_PREFIX

Type de données :
String

Propriété de :
ConnectionFactory, Destination

Lors de la création de rubriques temporaires, XMS génère une chaîne de rubrique au format "`TEMP/TEMPTOPICPREFIX/unique_id`" ou, si cette propriété contient la valeur par défaut, cette chaîne, "`TEMP/unique_id`", est générée. La spécification d'une valeur non vide permet la définition de files d'attente modèle spécifiques pour la création des files d'attente gérées pour les abonnés à des rubriques temporaires créées sous cette connexion.

Toute chaîne non nulle constituée de caractères valides pour une chaîne de rubrique IBM WebSphere MQ est une valeur valide pour cette propriété.

Par défaut, cette propriété est définie sur "" (chaîne vide).

Remarque : Cette propriété est pertinente uniquement dans le domaine publication/abonnement.

XMSC_WMQ_TEMPORARY_MODEL

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la file d'attente modèle WebSphere MQ à partir de laquelle une file d'attente dynamique est créée lorsque l'application crée une file d'attente temporaire Un XMS .

La valeur par défaut de la propriété est SYSTEM.DEFAULT.MODEL.QUEUE.

Cette propriété est pertinente uniquement dans le domaine point-à-point.

XMSC_WMQ_WILDCARD_FORMAT

Type de données :
System.Int32

Propriété de :
ConnectionFactory, Destination

Cette propriété détermine la version de syntaxe des caractères génériques à utiliser.

Lors de l'utilisation de la fonction de publication / abonnement avec IBM WebSphere MQ '*'et'?' sont traités comme des caractères génériques. Les caractères # et + sont traités comme des caractères

génériques lorsque vous utilisez le mode publication/abonnement avec IBM Integration Bus. Cette propriété remplace la propriété XMSC_WMQ_BROKER_VERSION.

Les valeurs valides pour cette propriété sont :

XMSC_WMQ_WILDCARD_TOPIC_ONLY
Reconnaît uniquement les caractères génériques de niveau rubrique, c'est-à-dire '#' et'+'sont traités comme des caractères génériques. Cette valeur est identique à XMSC_WMQ_BROKER_V2.

XMSC_WMQ_WILDCARD_CHAR_ONLY
Reconnaît uniquement les caractères génériques, c'est-à-dire '*' et'?' sont traités comme des caractères génériques. Cette valeur est identique à XMSC_WMQ_BROKER_V1.

Par défaut, cette propriété est définie sur XMSC_WMQ_WILDCARD_TOPIC_ONLY.

XMSC_WPM_BUS_NAME

Type de données :
String

Propriété de :
ConnectionFactory et Destination

Nom utilisé dans un URI :
busName

Pour une fabrique de connexions, nom du bus d'intégration de services auquel l'application se connecte ; pour une destination, nom du bus d'intégration de services dans lequel la destination existe.

Pour une destination sous forme de rubrique, cette propriété est le nom du bus d'intégration de services dans lequel existe l'espace de rubrique associé. Cet espace de rubrique est spécifié par la propriété XMSC_WPM_TOPIC_SPACE.

Si la propriété n'est pas définie pour une destination, la file d'attente ou l'espace de rubrique associé est supposé exister dans le bus d'intégration de services auquel l'application se connecte.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_CONNECTION_PROTOCOL

Type de données :
System.Int32

Propriété de :
Connexion

Protocole de communication utilisé pour la connexion au moteur de messagerie. Cette propriété est en lecture seule.

Les valeurs possibles sont les suivantes :

Valeur	Explication
XMSC_WPM_CP_HTTP	La connexion utilise HTTP sur TCP/IP.
XMSC_WPM_CP_TCP	La connexion utilise TCP/IP.

XMSC_WPM_CONNECTION_PROXIMITY

Type de données :
System.Int32

Propriété de :
ConnectionFactory

Paramètre de proximité de connexion pour la connexion. Cette propriété détermine quelle doit être la proximité entre le moteur de messagerie auquel l'application se connecte et le serveur d'amorçage.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Paramètre de proximité de connexion
XMSC_WPM_CONNECTION_PROXIMITY_BUS	Bus
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	Cluster
XMSC_WPM_CONNECTION_PROXIMITY_HOST	Hôte
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	serveur

La valeur par défaut est XMSC_WPM_CONNECTION_PROXIMITY_BUS.

XMSC_WPM_DUR_SUB_HOME

Type de données :
String

Propriété de :
ConnectionFactory

Nom utilisé dans un URI :
durableSubscriptionHome

Nom du moteur de messagerie où sont gérés les abonnements durables pour une connexion ou une destination. Le messages devant être distribués aux abonnés durables sont stockés sur le point de publication du même moteur de messagerie.

Un accueil d'abonnement durable doit être spécifié pour une connexion avant qu'une application puisse créer un abonné durable qui utilise la connexion. Toute valeur spécifiée pour une destination remplace la valeur spécifiée pour la connexion.

Par défaut, la propriété n'est pas définie.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WPM_HOST_NAME

Type de données :
String

Propriété de :
Connexion

Nom hôte ou adresse IP du système qui contient le moteur de messagerie auquel l'application est connectée. Cette propriété est en lecture seule.

XMSC_WPM_LOCAL_ADDRESS

Type de données :
String

Propriété de :
ConnectionFactory

Pour une connexion à un bus d'intégration de services, cette propriété spécifie l'interface de réseau local et/ou le port ou la plage de ports à utiliser.

La valeur de la propriété est une chaîne au format suivant :

[*nom_hôte*][(*port_bas*)[,*port_haut*])]

La signification des variables est la suivante :

nom_hôte
Nom d'hôte ou adresse IP de l'interface de réseau local à utiliser pour la connexion.

Cette information est requise uniquement si le système sur lequel l'application s'exécute dispose d'au moins deux interfaces réseau et que vous devez pouvoir spécifier l'interface qui doit être utilisée

pour la connexion. Si le système dispose d'une seule interface réseau, seule cette interface peut être utilisée. Si le système dispose d'au moins deux interfaces réseau et que vous ne spécifiez pas quelle interface doit être utilisée, celle-ci est sélectionnée au hasard.

port_bas
Numéro du port local à utiliser pour la connexion.

Si *port_haut* est aussi spécifiée, *port_bas* est interprétée comme correspondant au numéro de port le plus bas de la plage de ports.

port_haut
Numéro de port le plus élevé dans la plage de ports. Un des ports de la plage spécifiée doit être utilisé pour la connexion.

Voici quelques exemples de valeurs valides pour la propriété :

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Par défaut, la propriété n'est pas définie.

XMSC_WPM_ME_NAME

Type de données :
String

Propriété de :
Connexion

Nom du moteur de messagerie auquel l'application est connectée. Cette propriété est en lecture seule.

XMSC_WPM_NON_PERSISTENT_MAP

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Niveau de fiabilité des messages non persistants envoyés à l'aide de la connexion.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Niveau de fiabilité
XMSC_WPM_MAPPING_AS_DESTINATION	Déterminé par le niveau de fiabilité par défaut spécifié pour la file d'attente ou l'espace de rubrique dans le bus d'intégration de services
XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT	Non persistant - Meilleur effort
XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT	Non persistant - Express
XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT	Non persistant - Fiable

Valeur admise	Niveau de fiabilité
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Persistant - Fiable
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Persistant - Assuré

La valeur par défaut est XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT.

XMSC_WPM_PERSISTENT_MAP

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Niveau de fiabilité des messages non persistants envoyés à l'aide de la connexion.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Niveau de fiabilité
XMSC_WPM_MAPPING_AS_DESTINATION	Déterminé par le niveau de fiabilité par défaut spécifié pour la file d'attente ou l'espace de rubrique dans le bus d'intégration de services
XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT	Non persistant - Meilleur effort
XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT	Non persistant - Express
XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT	Non persistant - Fiable
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Persistant - Fiable
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Persistant - Assuré

La valeur par défaut est XMSC_WPM_MAPPING_RELIABLE_PERSISTENT.

XMSC_WPM_PORT

Type de données :
System.Int32

Propriété de :
Connexion

Numéro du port écouté par le moteur de messagerie auquel l'application est connectée. Cette propriété est en lecture seule.

XMSC_WPM_PROVIDER_ENDPOINTS

Type de données :
String

Propriété de :
ConnectionFactory

Séquence d'une ou plusieurs adresses de noeud final de serveurs d'amorçage. Les adresses de noeud final sont séparées par des virgules.

Un serveur d'amorçage est un serveur d'applications responsable de la sélection du moteur de messagerie auquel l'application se connecte. L'adresse de noeud final du serveur d'amorçage se présente sous la forme suivante :

nom_hôte:*numéro_port*:*nom_chaîne*

La signification des composants d'une adresse de noeud final est la suivante :

nom_hôte
Nom d'hôte ou adresse IP du système sur lequel se trouve le serveur d'amorçage. Si vous ne spécifiez aucun nom d'hôte ou aucune adresse IP, la valeur par défaut est `localhost`.

numéro_port
Numéro du port sur lequel le serveur d'amorçage est à l'écoute des demandes entrantes. Si vous ne spécifiez aucun numéro de port, la valeur par défaut est 7276.

nom_chaîne
Nom d'une chaîne de transport d'amorçage utilisée par le serveur d'amorçage. Les valeurs admises sont les suivantes :

Valeur admise	Nom de la chaîne de transport d'amorçage
XMSC_WPM_BOOTSTRAP_HTTP	BootstrapTunneledMessaging
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	BootstrapSecureMessaging
XMSC_WPM_BOOTSTRAP_TCP	BootstrapBasicMessaging

Si vous ne spécifiez aucun nom la valeur par défaut est XMSC_WPM_BOOTSTRAP_TCP.

Si vous ne spécifiez aucune adresse de noeud final, la valeur par défaut est localhost:7276:BootstrapBasicMessaging.

XMSC_WPM_SSL_CIPHER_SUITE

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la suite de chiffrement à utiliser pour une connexion TLS à un moteur de messagerie de bus d'intégration de services WebSphere. Le protocole utilisé lors de la négociation de la connexion sécurisée dépend de la suite de chiffrement spécifiée.

Tableau 35. Options CipherSuite pour la connexion à un moteur de messagerie bus d'intégration de services WebSphere

Algorithme de cryptographie	Protocole utilisé
TLS_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1

Remarques :

1. Les spécifications TLS_RSA_WITH_AES_128_CBC_SHA et TLS_RSA_WITH_AES_256_CBC_SHA CipherSuites sont prises en charge uniquement sur Windows ou Solaris. (Ceci est dicté par GSKit.)
2. TLS_RSA_WITH_3DES_EDE_CBC_SHA a été déprécié. Toutefois, vous pouvez tout de même l'utiliser pour transférer jusqu'à 32 Go de données avant que la connexion ne soit arrêtée avec l'erreur

AMQ9288. Pour éviter cette erreur, n'utilisez pas la norme DES triple ou activez la réinitialisation de clé confidentielle lors de l'utilisation de ce CipherSpec.

Cette propriété n'a pas de valeur par défaut. Si vous voulez utiliser SSL ou TLS, vous devez spécifier une valeur pour cette propriété, de sorte que votre application puisse se connecter au serveur.

XMSC_WPM_SSL_FIPS_REQUIRED

Type de données :
Boolean

Propriété de :
Fabrique de connexions

La valeur de cette propriété détermine si une application peut ou non utiliser des algorithme de cryptographie conformes non FIPS. Si cette propriété est définie à true, seuls des algorithmes FIPS sont utilisés pour la connexion client-serveur.La définition de la valeur de la propriété à TRUE empêche l'application d'utiliser des algorithmes de cryptographie conformes non FIPS.

Par défaut, la propriété est paramétrée sur FALSE (c'est-à-dire, avec le mode FIPS désactivé).

XMSC_WPM_SSL_KEY_REPOSITORY

Type de données :
String

Propriété de :
ConnectionFactory

Chemin d'accès au fichier de clés contenant les clés publiques ou privées à utiliser dans la connexion sécurisée.

La définition de la propriété de fichier de clés à la valeur spéciale de XMSC_WPM_SSL_MS_CERTIFICATE_STORE spécifie l'utilisation de la base de donnéesMicrosoft Windows. L'utilisation de la base de données de clés Microsoft Windows, disponible à partir de **Panneau de configuration** > **Options Internet** > **Contenu** > **Certificats**, dispense du besoin d'une base de données de clés distincte. L'utilisation de cette constante sur Windows x64 et d'autres plateformes n'est pas autorisée.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_SSL_KEYRING_LABEL

Type de données :
String

Propriété de :
ConnectionFactory

Certificat à utiliser lors de l'authentification avec le serveur. Si aucune valeur n'est spécifiée, le certificat par défaut est utilisé.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_SSL_KEYRING_PW

Type de données :
String

Propriété de :
ConnectionFactory

Mot de passe pour le fichier de clés.

Cette propriété peut être une alternative à XMSC_WPM_SSL_KEYRING_STASH_FILE pour configurer le mot de passe pour le fichier de clés.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_SSL_KEYRING_STASH_FILE

Type de données :
String

Propriété de :
ConnectionFactory

Nom d'un fichier binaire contenant le mot de passe du fichier référentiel principal.

Cette propriété peut être une alternative à XMSC_WPM_SSL_KEYRING_PW pour configurer le mot de passe pour le fichier de clés.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_TARGET_GROUP

Type de données :
String

Propriété de :
ConnectionFactory

Nom d'un groupe cible de moteurs de messagerie. La nature du groupe cible est déterminée par la propriété XMSC_WPM_TARGET_TYPE.

Définissez cette propriété si vous voulez restreindre la recherche d'un moteur de messagerie à un sous-groupe de moteurs du bus d'intégration de services. Si vous souhaitez que votre application puisse se connecter à tout moteur de messagerie du bus d'intégration de services, ne définissez pas cette propriété.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_TARGET_SIGNIFICANCE

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Signification du groupe cible de moteurs de messagerie.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WPM_TARGET_SIGNIFICANCE_ PREFERRED	Un moteur de messagerie du groupe cible est sélectionné, si l'un d'entre eux est disponible. Dans le cas contraire, un moteur de messagerie hors du groupe cible est sélectionné, sous réserve qu'il se trouve dans le même bus d'intégration de services.
XMSC_WPM_TARGET_SIGNIFICANCE_ REQUIRED	Le moteur de messagerie sélectionné doit être dans le groupe cible. Si aucun moteur de messagerie du groupe cible n'est disponible, le processus de connexion échoue.

La valeur par défaut de la propriété est XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED.

XMSC_WPM_TARGET_TRANSPORT_CHAIN

Type de données :
String

Propriété de :
ConnectionFactory

Nom de la chaîne de transport entrant que l'application doit utiliser pour se connecter à un moteur de messagerie.

La valeur de la propriété peut être le nom d'une chaîne de transport entrant disponible dans le serveur d'applications qui héberge le moteur de messagerie. La constante nommée suivante est fournie pour une des chaînes de transport entrant prédéfinies :

Constante nommée	Nom de la chaîne de transport
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	InboundBasicMessaging

La valeur par défaut de la propriété est XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC.

XMSC_WPM_TARGET_TYPE

Type de données :
System.Int32

Propriété de :
Fabrique de connexions

Type du groupe cible de moteurs de messagerie. Cette propriété détermine la nature du groupe cible identifié par la propriété XMSC_WPM_TARGET_GROUP.

Les valeurs admises de la propriété sont les suivantes :

Valeur admise	Explication
XMSC_WPM_TARGET_TYPE_BUSMEMBER	Le nom du groupe cible est le nom du membre de bus. Le groupe cible correspond à l'ensemble des moteurs de messagerie du membre de bus.
XMSC_WPM_TARGET_TYPE_CUSTOM	Le nom du groupe cible est le nom d'un groupe de moteurs de messagerie défini par l'utilisateur. Le groupe cible correspond à l'ensemble des moteurs de messagerie enregistrés avec le groupe défini par l'utilisateur.
XMSC_WPM_TARGET_TYPE_ME	Le nom du groupe cible est le nom d'un moteur de messagerie. Le groupe cible correspond au moteur de messagerie spécifié.

Par défaut, la propriété n'est pas définie.

XMSC_WPM_TEMP_Q_PREFIX

Type de données :
String

Propriété de :
ConnectionFactory

Préfixe utilisé pour former le nom de la file d'attente temporaire créée dans le bus d'intégration de services lorsque l'application crée une file d'attente temporaire Un XMS . Le préfixe peut contenir jusqu'à 12 caractères.

Le nom de la file d'attente temporaire commence par les caractères "_Q" suivis du préfixe. Le reste du nom est constitué de caractères générés par le système.

Par défaut, la propriété n'est pas définie, ce qui signifie que le nom d'une file d'attente temporaire n'a pas de préfixe.

Cette propriété est pertinente uniquement dans le domaine point-à-point.

XMSC_WPM_TEMP_TOPIC_PREFIX

Type de données :
String

Propriété de :
ConnectionFactory

Préfixe utilisé pour former le nom d'une rubrique temporaire créée par l'application. Le préfixe peut contenir jusqu'à 12 caractères.

Le nom d'une file d'attente temporaire commence par les caractères "_T" suivis du préfixe. Le reste du nom est constitué de caractères générés par le système.

Par défaut, la propriété n'est pas définie, ce qui signifie que le nom d'une rubrique temporaire n'a pas de préfixe.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

XMSC_WPM_TOPIC_SPACE

Type de données :
String

Propriété de :
Destination

Nom utilisé dans un URI :
topicSpace

Nom de l'espace de sujet contenant la rubrique. Seule une destination représentant une rubrique peut avoir cette propriété.

Par défaut, la propriété n'est pas définie, ce qui signifie que la rubrique par défaut est prise en charge.

Cette propriété est pertinente uniquement dans le domaine Publier/S'abonner.

Remarques

:NONE.

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 IBM 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 EMEA Director of Licensing
IBM Corporation
Tour Descartes
Armonk, NY 10504-1785
U.S.A.

Pour toute demande d'informations relatives au jeu de caractères codé sur deux octets, contactez le service de propriété intellectuelle IBM ou envoyez vos questions par courrier à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japon

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 CONTREFACON 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
Coordinateur d'interopérabilité logicielle, département 49XA
3605 Autoroute 52 N

Rochester, MN 55901
U.S.A.

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

Le logiciel sous licence décrit dans le présent document et tous les éléments sous disponibles s'y rapportant sont fournis par IBM conformément aux dispositions du Contrat sur les produits et services IBM, aux Conditions Internationales d'Utilisation de 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.

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 sur les droits d'auteur :

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.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Documentation sur l'interface de programmation

Les informations d'interface de programmation, si elles sont fournies, sont destinées à vous aider à créer un logiciel d'application à utiliser avec ce programme.

Ce manuel contient des informations sur les interfaces de programmation prévues qui permettent au client d'écrire des programmes pour obtenir les services de WebSphere MQ.

Toutefois, lesdites informations peuvent également contenir des données de diagnostic, de modification et d'optimisation. Ces données vous permettent de déboguer votre application.

Important : N'utilisez pas ces informations de diagnostic, de modification et d'optimisation en tant qu'interface de programmation car elles sont susceptibles d'être modifiées.

Marques

IBM, le logo IBM , ibm.com, sont des marques d' IBM Corporation dans de nombreux pays. La liste actualisée de toutes les marques d' IBM est disponible sur la page Web "Copyright and trademark

information"www.ibm.com/legal/copytrade.shtml. Les autres noms de produits et de services peuvent être des marques d'IBM ou d’autres sociétés.

Microsoft et Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans d'autres pays.

UNIX est une marque de The Open Group aux Etats-Unis et dans certains autres pays.

Linux® est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Ce produit inclut des logiciels développés par le projet Eclipse (http://www.eclipse.org/).

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques d'Oracle et/ou de ses sociétés affiliées.

Référence :

(1P) P/N: