7.5

Administration d' IBM WebSphere MQ

Remarque

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 169.

Cette édition s'applique à la version 7 édition 5 d' IBM® WebSphere MQ et à toutes les éditions et modifications ultérieures, sauf indication contraire dans les nouvelles éditions.

Table des matières

Administration de IBM WebSphere MQ

L'administration des gestionnaires de files d'attente et des ressources associées inclut les tâches que vous effectuez fréquemment pour activer et gérer ces ressources. Choisissez la méthode que vous préférez pour administrer vos gestionnaires de files d'attente et les ressources associées.

Vous pouvez administrer les objets IBM WebSphere MQ localement ou à distance (voir «Administration locale et à distance», à la page 7).

Il existe un certain nombre de méthodes différentes que vous pouvez utiliser pour créer et administrer vos gestionnaires de files d'attente et les ressources associées dans IBM WebSphere MQ. Ces méthodes incluent des interfaces de ligne de commande, une interface graphique et une API d'administration. Pour plus d'informations sur chacune de ces interfaces, voir les sections et les liens de cette rubrique.

Il existe différents ensembles de commandes que vous pouvez utiliser pour administrer IBM WebSphere MQ en fonction de votre plateforme:

- «IBM WebSphere MQ Les commandes de contrôle», à la page 5
- «Commandes IBM WebSphere MQ Script (MQSC)», à la page 5
- «Formats PCF (Programmable Command Formats)», à la page 6

Il existe également les autres options suivantes pour la création et la gestion des objets IBM WebSphere MQ :

- «Le IBM WebSphere MQ Explorer», à la page 6
- «Application de configuration par défaut de Windows», à la page 7
- «Le service Microsoft Cluster Service (MSCS)», à la page 7

Vous pouvez automatiser certaines tâches d'administration et de surveillance pour les gestionnaires de files d'attente locales et éloignées à l'aide des commandes PCF. Ces commandes peuvent également être simplifiées via l'utilisation de l'interface d'administration IBM WebSphere MQ (MQAI) sur certaines plateformes. Pour plus d'informations sur l'automatisation des tâches d'administration, voir «Automatisation des tâches d'administration», à la page 8.

IBM WebSphere MQ Les commandes de contrôle

Les commandes de contrôle vous permettent d'effectuer des tâches d'administration sur les gestionnaires de files d'attente eux-mêmes.

IBM WebSphere MQ pour Windows, les systèmes UNIX and Linux® fournissent les *commandes de contrôle* que vous émettez sur la ligne de commande du système.

Les commandes de contrôle sont décrites dans Création et gestion des gestionnaires de files d'attente. Pour obtenir les informations de référence sur les commandes de contrôle, voir Commandes de contrôleIBM WebSphere MQ.

Commandes IBM WebSphere MQ Script (MQSC)

Les commandes MQSC permettent de gérer les objets de gestionnaire de files d'attente, y compris le gestionnaire de files d'attente lui-même, les files d'attente, les définitions de processus, les listes de noms, les canaux, les canaux de connexion client, les programmes d'écoute, les services et les objets d'informations d'authentification.

Vous émettez des commandes MQSC vers un gestionnaire de files d'attente à l'aide de la commande `runmqsc` . Vous pouvez effectuer cette opération de manière interactive, en émettant des commandes à partir d'un clavier, ou vous pouvez rediriger le périphérique d'entrée standard (stdin) pour exécuter une séquence de commandes à partir d'un fichier texte ASCII. Dans les deux cas, le format des commandes est identique.

Vous pouvez exécuter la commande `runmqsc` dans trois modes, en fonction des indicateurs définis dans la commande:

- *Mode de vérification*, dans lequel les commandes MQSC sont vérifiées sur un gestionnaire de files d'attente local, mais ne sont pas exécutées
- *Mode direct*, dans lequel les commandes MQSC sont exécutées sur un gestionnaire de files d'attente local
- *Mode indirect*, dans lequel les commandes MQSC sont exécutées sur un gestionnaire de files d'attente éloignées

Les attributs d'objet spécifiés dans les commandes MQSC sont affichés dans cette section en majuscules (par exemple, RQMNAME), bien qu'ils ne soient pas sensibles à la casse. Les noms d'attribut de commande MQSC sont limités à huit caractères.

Les commandes MQSC sont disponibles sur toutes les plateformes. Les commandes MQSC sont récapitulées dans Comparaison des ensembles de commandes.

Sous Windows, UNIX ou Linux, vous pouvez utiliser le MQSC en tant que commandes uniques émises sur la ligne de commande du système. Pour émettre des commandes plus complexes ou multiples, le MQSC peut être intégré dans un fichier que vous exécutez à partir de la ligne de commande système Windows, UNIX ou Linux . MQSC peut être envoyé à un gestionnaire de files d'attente éloignées. Pour plus de détails, voir Référence MQSC.

Le «Commandes de script (MQSC)», à la page 76 contient une description de chaque commande MQSC et de sa syntaxe.

Voir «Exécution de tâches d'administration locales à l'aide des commandes MQSC», à la page 76 pour plus d'informations sur l'utilisation des commandes MQSC dans l'administration locale.

Formats PCF (Programmable Command Formats)

Les formats PCF (Programmable Command Formats) définissent les messages de commande et de réponse qui peuvent être échangés entre un programme et n'importe quel gestionnaire de files d'attente (qui prend en charge les PCF) d'un réseau. Vous pouvez utiliser des commandes PCF dans un programme d'application de gestion des systèmes pour l'administration des objets IBM WebSphere MQ : objets d'informations d'authentification, canaux, programmes d'écoute de canal, listes de noms, définitions de processus, gestionnaires de files d'attente, files d'attente, services et classes de stockage. L'application peut fonctionner à partir d'un point unique du réseau pour communiquer des informations de commande et de réponse avec n'importe quel gestionnaire de files d'attente, local ou distant, à l'aide du gestionnaire de files d'attente local.

Pour plus d'informations sur les fichiers PCF, voir «Présentation des commandes PCF (Programmable Command Formats)», à la page 9.

Pour la définition des fichiers PCF et des structures des commandes et des réponses, voir Programmable command formats reference.

Le IBM WebSphere MQ Explorer

A l'aide de l' IBM WebSphere MQ Explorer, vous pouvez effectuer les actions suivantes:

- Définissez et contrôlez diverses ressources, notamment les gestionnaires de files d'attente, les files d'attente, les définitions de processus, les listes de noms, les canaux, les canaux de connexion client, les programmes d'écoute, les services et les clusters.
- Démarrez ou arrêtez un gestionnaire de files d'attente local et ses processus associés.
- Affichez les gestionnaires de files d'attente et les objets associés sur votre poste de travail ou à partir d'autres postes de travail.
- Vérifiez le statut des gestionnaires de files d'attente, des clusters et des canaux.
- Vérifiez les applications, les utilisateurs ou les canaux pour lesquels une file d'attente particulière est ouverte, à partir du statut de la file d'attente.

Sur les systèmes Windows et Linux , vous pouvez démarrer IBM WebSphere MQ Explorer à l'aide du menu système, du fichier exécutable `MQExplorer` ou de la commande **strmqcfg** .

Sous Linux, pour démarrer correctement IBM WebSphere MQ Explorer , vous devez pouvoir écrire un fichier dans votre répertoire de base et ce dernier doit exister.

Vous pouvez utiliser IBM WebSphere MQ Explorer pour administrer les gestionnaires de files d'attente éloignées sur d'autres plateformes, y compris z/OS, pour plus de détails et pour télécharger SupportPac MS0T, voir https://www.ibm.com/support/docview.wss?uid=swg24021041.

Pour plus d'informations, voir «Administration à l'aide de IBM WebSphere MQ Explorer», à la page 58.

Application de configuration par défaut de Windows

Vous pouvez utiliser le programme de configuration par défaut Windows pour créer un ensemble d'objets IBM WebSphere MQ *starter* (ou par défaut). Un récapitulatif des objets par défaut créés est répertorié dans le Tableau 1: Objets créés par l'application de configuration par défaut Windows.

Le service Microsoft Cluster Service (MSCS)

Microsoft Cluster Service (MSCS) vous permet de connecter des serveurs à un *cluster*, ce qui offre une plus grande disponibilité des données et des applications et facilite la gestion du système. MSCS détecte automatiquement les échecs de serveurs ou d'applications et initie alors les restaurations requises.

Il est important de ne pas confondre les clusters au sens MSCS avec les clusters IBM WebSphere MQ . La distinction est la suivante:

IBM WebSphere MQ clusters
: sont des groupes de deux ou plusieurs gestionnaires de files d'attente sur un ou plusieurs ordinateurs, assurant une interconnexion automatique et permettant le partage de files d'attente entre eux pour l'équilibrage de charge et la redondance.

Clusters MSCS
: Des groupes d'ordinateurs, connectés ensemble et configurés de telle sorte que, en cas de défaillance, MSCS effectue une *reprise en ligne*, en transférant les données d'état des applications de l'ordinateur défaillant vers un autre ordinateur du cluster et en réexécutant leur fonctionnement.

La prise en charge de Microsoft Cluster Service (MSCS) fournit des informations détaillées sur la configuration de votre système IBM WebSphere MQ for Windows pour l'utilisation de MSCS.

Concepts associés

Présentation technique de WebSphere MQ

«Administration des objets IBM WebSphere MQ locaux», à la page 71
Cette section explique comment administrer les objets IBM WebSphere MQ locaux pour prendre en charge les programmes d'application qui utilisent l'interface MQI (Message Queue Interface). Dans ce contexte, l'administration locale implique la création, l'affichage, la modification, la copie et la suppression d'objets IBM WebSphere MQ .

«Administration des objets IBM WebSphere MQ distants», à la page 108

Considérations à prendre en compte en cas de perte du contact avec le gestionnaire de ressources XA

Tâches associées

Planification

Configuration

Référence associée

Scénarios de support transactionnel

Administration locale et à distance

Vous pouvez administrer des objets WebSphere MQ localement ou à distance.

Administration locale signifie l'exécution de tâches d'administration sur tous les gestionnaires de files d'attente que vous avez définis sur votre système local. Vous pouvez accéder à d'autres systèmes, par exemple via le programme d'émulation de terminal TCP/IP **telnet**, et y effectuer l'administration. Dans WebSphere MQ, vous pouvez considérer cela comme une administration locale car aucun canal n'est impliqué, c'est-à-dire que la communication est gérée par le système d'exploitation.

WebSphere MQ prend en charge l'administration à partir d'un point de contact unique via ce que l'on appelle l' *administration à distance*. Cela vous permet d'émettre des commandes à partir de votre système local qui sont traitées sur un autre système et s'applique également à WebSphere MQ Explorer. Par exemple, vous pouvez émettre une commande distante pour modifier une définition de file d'attente sur un gestionnaire de files d'attente éloignées. Vous n'avez pas besoin de vous connecter à ce système, bien que les canaux appropriés doivent être définis. Le gestionnaire de files d'attente et le serveur de commandes sur le système cible doivent être en cours d'exécution.

Certaines commandes ne peuvent pas être émises de cette manière, notamment la création ou le démarrage de gestionnaires de files d'attente et de serveurs de commandes. Pour effectuer ce type de tâche, vous devez vous connecter au système distant et émettre les commandes à partir de celui-ci ou créer un processus qui peut émettre les commandes pour vous. Cette restriction s'applique également à WebSphere MQ Explorer.

«Administration des objets IBM WebSphere MQ distants», à la page 108 décrit plus en détail le sujet de l'administration à distance.

Utilisation des commandes de contrôle IBM WebSphere MQ

Cette section explique comment utiliser les commandes de contrôle IBM WebSphere MQ .

Si vous souhaitez émettre des commandes de contrôle, votre ID utilisateur doit être membre du groupe mqm. Pour plus d'informations à ce sujet, voir Droits d'administration de WebSphere MQ sur les systèmes UNIX, Linux et Windows . En outre, notez les informations suivantes spécifiques à l'environnement:

WebSphere MQ pour Windows
Toutes les commandes de contrôle peuvent être émises à partir d'une ligne de commande. Les noms de commande et leurs indicateurs ne sont pas sensibles à la casse: vous pouvez les entrer en majuscules, en minuscules ou dans une combinaison de majuscules et de minuscules. Toutefois, les arguments des commandes de contrôle (tels que les noms de file d'attente) sont sensibles à la casse.

Dans les descriptions de syntaxe, le trait d'union (-) est utilisé comme indicateur d'indicateur. Vous pouvez utiliser la barre oblique (/) à la place du trait d'union.

Systèmes WebSphere MQ for UNIX and Linux
Toutes les commandes de contrôle WebSphere MQ peuvent être émises à partir d'un shell. Toutes les commandes sont sensibles à la casse.

Un sous-ensemble des commandes de contrôle peut être émis à l'aide de l'explorateur IBM WebSphere MQ .

Pour plus d'informations, voir Les commandes de contrôle WebSphere MQ

Automatisation des tâches d'administration

Vous pouvez décider qu'il serait avantageux pour votre installation d'automatiser certaines tâches d'administration et de surveillance. Vous pouvez automatiser les tâches d'administration pour les gestionnaires de files d'attente locales et éloignées à l'aide des commandes PCF (Programmable Command Format). Cette section suppose que vous avez l'expérience de l'administration des objets WebSphere MQ .

Commandes PCF

Les commandes PCF (Programmable Command Format) WebSphere MQ peuvent être utilisées pour programmer des tâches d'administration dans un programme d'administration. Ainsi, à partir d'un programme, vous pouvez manipuler des objets de gestionnaire de files d'attente (files d'attente,

définitions de processus, listes de noms, canaux, canaux de connexion client, programmes d'écoute, services et objets d'informations d'authentification) et même manipuler les gestionnaires de files d'attente eux-mêmes.

Les commandes PCF couvrent la même gamme de fonctions que les commandes MQSC. Vous pouvez écrire un programme pour émettre des commandes PCF vers n'importe quel gestionnaire de files d'attente du réseau à partir d'un noeud unique. De cette manière, vous pouvez à la fois centraliser et automatiser les tâches d'administration.

Chaque commande PCF est une structure de données imbriquée dans la partie données d'application d'un message WebSphere MQ . Chaque commande est envoyée au gestionnaire de files d'attente cible à l'aide de la fonction MQI MQPUT de la même manière que tout autre message. A condition que le serveur de commandes soit en cours d'exécution sur le gestionnaire de files d'attente qui reçoit le message, le serveur de commandes l'interprète comme un message de commande et exécute la commande. Pour obtenir les réponses, l'application émet un appel MQGET et les données de réponse sont renvoyées dans une autre structure de données. L'application peut ensuite traiter la réponse et agir en conséquence.

Remarque : Contrairement aux commandes MQSC, les commandes PCF et leurs réponses ne sont pas au format texte que vous pouvez lire.

En bref, voici quelques-uns des éléments nécessaires à la création d'un message de commande PCF:

Descripteur de message
Il s'agit d'un descripteur de message WebSphere MQ standard, dans lequel:

- Le type de message (*MsqType*) est MQMT_REQUEST.
- Le format de message (*Format*) est MQFMT_ADMIN.

Données d'application
Contient le message PCF incluant l'en-tête PCF, dans lequel:

- Le type de message PCF (*Type*) indique MQCFT_COMMAND.
- L'identificateur de la commande indique la commande, par exemple *Change Queue* (MQCMD_CHANGE_Q).

Pour obtenir une description complète des structures de données PCF et savoir comment les implémenter, voir «Présentation des commandes PCF (Programmable Command Formats)», à la page 9.

Attributs d'objet PCF

Les attributs d'objet dans PCF ne sont pas limités à huit caractères, comme pour les commandes MQSC. Ils sont présentés dans ce guide en italique. Par exemple, l'équivalent PCF de RQMNAME est *RemoteQMgrName*.

Echapper les fichiers PCF

Les fichiers PCF d'arrêt programme sont des commandes PCF qui contiennent des commandes MQSC dans le texte du message. Vous pouvez utiliser des fichiers PCF pour envoyer des commandes à un gestionnaire de files d'attente éloignées. Pour plus d'informations sur les fichiers PCF d'échappement, voir Echap.

Présentation des commandes PCF (Programmable Command Formats)

Les formats PCF (Programmable Command Formats) définissent les messages de commande et de réponse qui peuvent être échangés entre un programme et n'importe quel gestionnaire de files d'attente (qui prend en charge les PCF) d'un réseau. Les fichiers PCF simplifient l'administration du gestionnaire de files d'attente et d'autres fonctions d'administration du réseau. Ils peuvent être utilisés pour résoudre le problème de l'administration complexe des réseaux distribués, en particulier à mesure que les réseaux augmentent en taille et en complexité.

Les formats de commande programmables décrits dans cette documentation sont pris en charge par:

- IBM WebSphere MQ for AIX
- IBM WebSphere MQ for HP-UX
- IBM WebSphere MQ for Linux
- IBM WebSphere MQ for Solaris
- IBM WebSphere MQ for Windows
- IBM WebSphere MQ for HP Integrity NonStop Server

Résolution des incidents liés aux commandes PCF

L'administration des réseaux distribués peut devenir complexe. Les problèmes d'administration continuent de croître à mesure que les réseaux augmentent en taille et en complexité.

Exemples d'administration spécifique à la messagerie et à la mise en file d'attente:

- Gestion des ressources.

  Par exemple, la création et la suppression de files d'attente.
- Surveillance des performances.

  Par exemple, la longueur maximale de la file d'attente ou le débit des messages.
- Contrôle.

  Par exemple, l'optimisation des paramètres de file d'attente tels que la longueur maximale de la file d'attente, la longueur maximale des messages et l'activation et la désactivation des files d'attente.
- Routage des messages.

  Définition de routes alternatives à travers un réseau.

WebSphere MQ Les commandes PCF peuvent être utilisées pour simplifier l'administration du gestionnaire de files d'attente et d'autres fonctions d'administration du réseau. Les commandes PCF vous permettent d'utiliser une application unique pour effectuer l'administration du réseau à partir d'un seul gestionnaire de files d'attente au sein du réseau.

Que sont les PCF?

Les PCF définissent les messages de commande et de réponse qui peuvent être échangés entre un programme et n'importe quel gestionnaire de files d'attente (qui prend en charge les PCF) d'un réseau. Vous pouvez utiliser des commandes PCF dans un programme d'application de gestion des systèmes pour l'administration des objets WebSphere MQ : objets d'informations d'authentification, canaux, programmes d'écoute de canal, listes de noms, définitions de processus, gestionnaires de files d'attente, files d'attente, services et classes de stockage. L'application peut fonctionner à partir d'un point unique du réseau pour communiquer des informations de commande et de réponse avec n'importe quel gestionnaire de files d'attente, local ou distant, à l'aide du gestionnaire de files d'attente local.

Chaque gestionnaire de files d'attente possède une file d'attente d'administration avec un nom de file d'attente standard et votre application peut envoyer des messages de commande PCF à cette file d'attente. Chaque gestionnaire de files d'attente dispose également d'un serveur de commandes pour traiter les messages de commande de la file d'attente d'administration. Les messages de commande PCF peuvent donc être traités par n'importe quel gestionnaire de files d'attente du réseau et les données de réponse peuvent être renvoyées à votre application à l'aide de la file d'attente de réponses spécifiée. Les commandes PCF et les messages de réponse sont envoyés et reçus à l'aide de l'interface MQI (Message Queue Interface) normale.

Pour obtenir la liste des commandes PCF disponibles, y compris leurs paramètres, voir Définitions des formats de commande programmables.

Utilisation des formats de commande programmables

Vous pouvez utiliser des fichiers PCF dans un programme de gestion des systèmes pour l'administration à distance de WebSphere MQ .

Cette section contient :

Messages de commande PCF

Les messages de commande PCF sont constitués d'un en-tête PCF, de paramètres identifiés dans cet en-tête et de données de message définies par l'utilisateur. Les messages sont émis à l'aide d'appels d'interface de file d'attente de messages.

Chaque commande et ses paramètres sont envoyés sous la forme d'un message de commande distinct contenant un en-tête PCF suivi d'un certain nombre de structures de paramètres ; pour plus de détails sur l'en-tête PCF, voir MQCFH-PCF headeret pour un exemple de structure de paramètre, voir MQCFST-PCF string parameter. L'en-tête PCF identifie la commande et le nombre de structures de paramètres qui suivent dans le même message. Chaque structure de paramètres fournit un paramètre à la commande.

Les réponses aux commandes, générées par le serveur de commandes, ont une structure similaire. Il existe un en-tête PCF, suivi d'un certain nombre de structures de paramètres. Les réponses peuvent consister en plusieurs messages, mais les commandes consistent toujours en un seul message.

Sur les plateformes autres que z/OS, la file d'attente à laquelle les commandes PCF sont envoyées est toujours appelée SYSTEM.ADMIN.COMMAND.QUEUE.

Comment émettre des messages de commande PCF

Utilisez les appels MQI (Message Queue Interface) normaux, MQPUT, MQGET, etc., pour insérer et extraire des messages de commande et de réponse PCF vers et depuis leurs files d'attente.

Remarque :

Vérifiez que le serveur de commandes est en cours d'exécution sur le gestionnaire de files d'attente cible pour que la commande PCF puisse être traitée sur ce gestionnaire de files d'attente.

Pour obtenir la liste des fichiers d'en-tête fournis, voir WebSphere MQ COPY, header, include et module files.

Descripteur de message pour une commande PCF

Le descripteur de message WebSphere MQ est intégralement documenté dans MQMD-Descripteur de message.

Un message de commande PCF contient les zones suivantes dans le descripteur de message:

Report
Toute valeur valide, selon les besoins.

MsgType
Cette zone doit être MQMT_REQUEST pour indiquer un message nécessitant une réponse.

Expiry
Toute valeur valide, selon les besoins.

Feedback
Défini sur MQFB_NONE

Encoding
Si vous envoyez des données à des systèmes Windows, UNIX ou Linux , définissez cette zone sur le codage utilisé pour les données de message ; la conversion est effectuée si nécessaire.

CodedCharSetId
Si vous envoyez àWindows, systèmes UNIX ou Linux , définissez cette zone sur l'identificateur de jeu de caractères codés utilisé pour les données de message ; la conversion est effectuée si nécessaire.

Format
Défini sur MQFMT_ADMIN.

Priority
Toute valeur valide, selon les besoins.

Persistence
Toute valeur valide, selon les besoins.

MsgId
L'application émettrice peut spécifier n'importe quelle valeur ou MQMI_NONE peut être spécifié pour demander au gestionnaire de files d'attente de générer un identificateur de message unique.

CorrelId
L'application émettrice peut spécifier n'importe quelle valeur ou MQCI_NONE peut être spécifié pour indiquer qu'il n'y a pas d'identificateur de corrélation.

ReplyToQ
Nom de la file d'attente de réception de la réponse.

ReplyToQMgr
Nom du gestionnaire de files d'attente pour la réponse (ou vide).

Zones de contexte de message
Ces zones peuvent être définies sur n'importe quelle valeur valide, selon les besoins. Normalement, l'option d'insertion de message MQPMO_DEFAULT_CONTEXT est utilisée pour définir les zones de contexte de message sur les valeurs par défaut.

Si vous utilisez une structure MQMD version-2 , vous devez définir les zones supplémentaires suivantes:

GroupId
Défini sur MQGI_NONE

MsgSeqNumber
Définir sur 1

Offset
Définir sur 0

MsgFlags
Défini sur MQMF_NONE

OriginalLength
Défini sur MQOL_UNDEFINED

Envoi de données utilisateur

Les structures PCF peuvent également être utilisées pour envoyer des données de message définies par l'utilisateur. Dans ce cas, la zone *Format* du descripteur de message doit être définie sur MQFMT_PCF.

Envoi et réception de messages PCF dans une file d'attente spécifiée

Envoi de messages PCF à une file d'attente spécifiée

Pour envoyer un message à une file d'attente spécifiée, l'appel de sac mqPutconvertit le contenu du sac spécifié en message PCF et envoie le message à la file d'attente spécifiée. Le contenu du sac reste inchangé après l'appel.

Comme entrée pour cet appel, vous devez fournir:

- Un descripteur de connexion MQI.
- Descripteur d'objet de la file d'attente dans laquelle le message doit être placé.

- Descripteur de message. Pour plus d'informations sur le descripteur de message, voir MQMD-Descripteur de message.
- Options d'insertion de message à l'aide de la structure MQPMO. Pour plus d'informations sur la structure MQPMO, voir MQPMO-Options d'insertion de message.
- Poignée du sac à convertir en message.

  Remarque : Si le sac contient un message d'administration et que l'appel d'interrogation mqAdda été utilisé pour insérer des valeurs dans le sac, la valeur de l'élément de données MQIASY_COMMAND doit être une commande INQUIRE reconnue par MQAI.

Pour une description complète de l'appel de sac mqPut, voir mqPutBag.

Réception de messages PCF d'une file d'attente spécifiée

Pour recevoir un message d'une file d'attente spécifiée, l'appel mqGetBag extrait un message PCF d'une file d'attente spécifiée et convertit les données de message en un sac de données.

Comme entrée pour cet appel, vous devez fournir:

- Un descripteur de connexion MQI.
- Descripteur d'objet de la file d'attente à partir de laquelle le message doit être lu.
- Descripteur de message. Dans la structure MQMD, le paramètre `Format` doit être MQFMT_ADMIN, MQFMT_EVENT ou MQFMT_PCF.

  Remarque : Si le message est reçu dans une unité de travail (c'est-à-dire avec l'option MQGMO_SYNCPOINT) et que le format du message n'est pas pris en charge, l'unité de travail peut être annulée. Le message est ensuite réintégré dans la file d'attente et peut être extrait à l'aide de l'appel MQGET au lieu de l'appel Bag mqGet. Pour plus d'informations sur le descripteur de message, voir Options MQGMO-Get-message.
- Obtenir les options de message à l'aide de la structure MQGMO. Pour plus d'informations sur la structure MQGMO, voir MQMD-Descripteur de message.
- Poignée du sac devant contenir le message converti.

Pour une description complète de l'appel de sac mqGet, voir mqGetBag.

Réponses

En réponse à chaque commande, le serveur de commandes génère un ou plusieurs messages de réponse. Un message de réponse a un format similaire à celui d'un message de commande.

L'en-tête PCF possède la même valeur d'identificateur de commande que la commande à laquelle il correspond à une réponse (voir MQCFH-en-tête PCF pour plus de détails). L'identificateur de message et l'identificateur de corrélation sont définis en fonction des options de rapport de la demande.

Si le type d'en-tête PCF du message de commande est MQCFT_COMMAND, seules les réponses standard sont générées. Ces commandes sont prises en charge sur toutes les plateformes, à l'exception de z/OS. Les anciennes applications ne prennent pas en charge PCF sur z/OS; WebSphere MQ Windows Explorer est l'une de ces applications (toutefois, la version 6.0 ou ultérieure IBM WebSphere MQ Explorer prend en charge PCF sur z/OS).

Si le type d'en-tête PCF du message de commande est MQCFT_COMMAND_XR, des réponses étendues ou standard sont générées. Ces commandes sont prises en charge sur z/OS et sur d'autres plateformes. Les commandes émises sur z/OS génèrent uniquement des réponses étendues. Sur d'autres plateformes, l'un ou l'autre type de réponse peut être généré.

Si une seule commande spécifie un nom d'objet générique, une réponse distincte est renvoyée dans son propre message pour chaque objet correspondant. Pour la génération de réponse, une seule commande avec un nom générique est traitée comme plusieurs commandes individuelles (à l'exception de la zone de contrôle MQCFC_LAST ou MQCFC_NOT_LAST). Sinon, un message de commande génère un message de réponse.

Certaines réponses PCF peuvent renvoyer une structure même si elle n'est pas demandée. Cette structure apparaît dans la définition de la réponse (Définitions des formats de commande programmables) comme *toujours renvoyé*. La raison pour laquelle, pour ces réponses, il est nécessaire de nommer les objets dans la réponse pour identifier l'objet auquel les données s'appliquent.

Descripteur de message pour une réponse

Un message de réponse comporte les zones suivantes dans le descripteur de message:

MsgType
Cette zone est MQMT_REPLY.

MsgId
Cette zone est générée par le gestionnaire de files d'attente.

CorrelId
Cette zone est générée en fonction des options de rapport du message de commande.

Format
Cette zone est MQFMT_ADMIN.

Encoding
Défini sur MQENC_NATIVE.

CodedCharSetId
Défini sur MQCCSI_Q_MGR.

Persistence
Identique à celui du message de commande.

Priority
Identique à celui du message de commande.

La réponse est générée avec MQPMO_PASS_IDENTITY_CONTEXT.

Réponses standard

Les messages de commande avec un type d'en-tête MQCFT_COMMAND, des réponses standard sont générées. Ces commandes sont prises en charge sur toutes les plateformes, à l'exception de z/OS.

Il existe trois types de réponse standard:

- Réponse OK
- Réponse d'erreur
- Réponse aux données

Réponse OK

Cette réponse se compose d'un message commençant par un en-tête de format de commande, avec une zone *CompCode* de MQCC_OK ou MQCC_WARNING.

Pour MQCC_OK, *Reason* est MQRC_NONE.

Pour MQCC_WARNING, *Reason* identifie la nature de l'avertissement. Dans ce cas, l'en-tête de format de commande peut être suivi d'une ou de plusieurs structures de paramètre d'avertissement appropriées à ce code anomalie.

Dans les deux cas, pour une commande d'interrogation, d'autres structures de paramètres peuvent suivre, comme décrit dans les sections suivantes.

Réponse d'erreur

Si la commande comporte une erreur, un ou plusieurs messages de réponse d'erreur sont envoyés (plusieurs peuvent être envoyés même pour une commande qui n'aurait normalement qu'un seul message de réponse). Ces messages de réponse d'erreur ont MQCFC_LAST ou MQCFC_NOT_LAST définis comme approprié.

Chaque message de ce type commence par un en-tête de format de réponse, avec une valeur *CompCode* de MQCC_FAILED et une zone *Reason* qui identifie l'erreur particulière. En général, chaque message décrit une erreur différente. En outre, chaque message a soit zéro, soit une (jamais plus d'une) structure de paramètres d'erreur après l'en-tête. Cette structure de paramètre, s'il en existe une, est une structure MQCFIN, avec une zone *Parameter* contenant l'un des éléments suivants:

- ID_PARAMÈTRE_MQIACF_ID

  La zone *Value* de la structure correspond à l'identificateur du paramètre erroné (par exemple, MQCA_Q_NAME).

- ID_ERREUR_MQIACF

  Cette valeur est utilisée avec une valeur *Reason* (dans l'en-tête de format de commande) de MQRC_UNEXPECTED_ERROR. La zone *Value* de la structure MQCFIN correspond au code anomalie inattendu reçu par le serveur de commandes.

- MQIACF_SELECTOR

  Cette valeur se produit si une structure de liste (MQCFIL) envoyée avec la commande contient un sélecteur en double ou non valide. La zone *Reason* dans l'en-tête de format de commande identifie l'erreur et la zone *Value* dans la structure MQCFIN est la valeur de paramètre dans la structure MQCFIL de la commande qui était erronée.

- MQIACF_DÉCALAGE_ERREUR_ERREUR

  Cette valeur se produit en cas d'erreur de comparaison de données dans la commande Ping Channel. La zone *Value* de la structure correspond au décalage de l'erreur de comparaison du canal ping.

- MQIA_CODED_CHAR_SET_ID

  Cette valeur se produit lorsque l'identificateur de jeu de caractères codés dans le descripteur de message du message de commande PCF entrant ne correspond pas à celui du gestionnaire de files d'attente cible. La zone *Value* de la structure correspond à l'identificateur de jeu de caractères codés du gestionnaire de files d'attente.

Le dernier (ou seul) message de réponse d'erreur est une réponse récapitulative, avec une zone *CompCode* de MQCC_FAILED et une zone *Reason* de MQRCCF_COMMAND_FAILED. Ce message n'a pas de structure de paramètre à la suite de l'en-tête.

Réponse aux données

Cette réponse consiste en une réponse OK (comme décrit précédemment) à une commande d'interrogation. La réponse OK est suivie de structures supplémentaires contenant les données demandées, comme décrit dans la rubrique Définitions des formats de commande programmables.

Les applications ne doivent pas dépendre du renvoi de ces structures de paramètres supplémentaires dans un ordre particulier.

Contrôle des droits d'accès pour les commandes PCF

Lorsqu'une commande PCF est traitée, le *UserIdentifier* du descripteur de message dans le message de commande est utilisé pour les vérifications des droits d'accès aux objets WebSphere MQ requises. La vérification des droits d'accès est implémentée différemment sur chaque plateforme, comme décrit dans cette rubrique.

Les vérifications sont effectuées sur le système sur lequel la commande est en cours de traitement ; par conséquent, cet ID utilisateur doit exister sur le système cible et disposer des droits requis pour traiter la commande. Si le message provient d'un système distant, l'un des moyens d'obtenir l'ID existant sur le système cible consiste à disposer d'un ID utilisateur correspondant sur les systèmes local et distant.

IBM WebSphere MQ pour Windows, systèmes UNIX and Linux

Pour traiter une commande PCF, l'ID utilisateur doit disposer des droits *dsp* sur l'objet gestionnaire de files d'attente sur le système cible. En outre, des vérifications des droits sur les objets WebSphere MQ sont effectuées pour certaines commandes PCF, comme illustré dans la Tableau 1, à la page 17.

Pour traiter l'une des commandes suivantes , l'ID utilisateur doit appartenir au groupe *mqm*.

Remarque : Pour Windows **uniquement**, l'ID utilisateur peut appartenir au groupe *Administrateurs* ou au groupe *mqm*.

- Modifier un canal
- Copier un canal
- Créer un canal
- Supprimer un canal
- Envoyer une commande PING à un canal
- Réinitialisation du canal
- Démarrer un canal
- Arrêter le canal
- Démarrer un initialisateur de canal
- Démarrer un programme d'écoute de canaux
- Résolution du canal
- Réinitialisation d'un cluster
- Régénérer un cluster
- Interrompre un gestionnaire de files d'attente
- Reprendre un gestionnaire de files d'attente

WebSphere MQ pour HP Integrity NonStop Server

Pour traiter une commande PCF, l'ID utilisateur doit disposer des droits *dsp* sur l'objet gestionnaire de files d'attente sur le système cible. En outre, des vérifications des droits d'accès aux objets IBM WebSphere MQ sont effectuées pour certaines commandes PCF, comme illustré dans la Tableau 1, à la page 17.

Pour traiter l'une des commandes suivantes , l'ID utilisateur doit appartenir au groupe *mqm*:

- Modifier un canal
- Copier un canal
- Créer un canal
- Supprimer un canal
- Envoyer une commande PING à un canal
- Réinitialisation du canal
- Démarrer un canal
- Arrêter le canal
- Démarrer un initialisateur de canal
- Démarrer un programme d'écoute de canaux
- Résolution du canal
- Réinitialisation d'un cluster
- Régénérer un cluster
- Interrompre un gestionnaire de files d'attente
- Reprendre un gestionnaire de files d'attente

Droits sur les objets WebSphere MQ

Tableau 1. Windows, HP Integrity NonStop Server, systèmes UNIX and Linux -droits sur les objets		
Commande	**Droits sur les objets WebSphere MQ**	**Droits sur les classes (pour le type d'objet)**
Modifier des informations d'authentification	dsp et chg	Non applicable
Modifier un canal	dsp et chg	Non applicable
Modifier le programme d'écoute de canal	dsp et chg	Non applicable
Modifier le canal de connexion client	dsp et chg	Non applicable
Modifier une liste de noms	dsp et chg	Non applicable
Changement de processus	dsp et chg	Non applicable
Modifier une file d'attente	dsp et chg	Non applicable
Modifier un gestionnaire de files d'attente	chg *voir les remarques 3 et 5*	Non applicable
Modifier le service	dsp et chg	Non applicable
Mettre à blanc une file d'attente	clr	Non applicable
Copier des informations d'authentification	dsp	crt
Copier les informations d'authentification (remplacement) *voir la remarque 1*	*de:* dsp *à:* chg	crt
Copier un canal	dsp	crt
Copier le canal (remplacer) *voir la remarque 1*	*de:* dsp *à:* chg	crt
Programme d'écoute de canal de copie	dsp	crt
Programme d'écoute de canal de copie (remplacement) *voir la remarque 1*	*de:* dsp *à:* chg	crt
Copier le canal de connexion client	dsp	crt
Copier le canal de connexion client (remplacement) *voir la remarque 1*	*de:* dsp *à:* chg	crt
Copier une liste de noms	dsp	crt
Copier la liste de noms (remplacement) *voir la remarque 1*	*de:* dsp *à:* dsp et chg	crt
Copier un processus	dsp	crt

Tableau 1. Windows, HP Integrity NonStop Server, systèmes UNIX and Linux -droits sur les objets (suite)		
Commande	**Droits sur les objets WebSphere MQ**	**Droits sur les classes (pour le type d'objet)**
Processus de copie (remplacement) *voir la remarque 1*	*de:* dsp *à:* chg	crt
Copier une file d'attente	dsp	crt
File d'attente de copie (remplacement) *voir la remarque 1*	*de:* dsp *à:* dsp et chg	crt
Créer des informations d'authentification	*(informations d'authentification par défaut du système)* dsp	crt
Créer des informations d'authentification (remplacer) *voir la remarque 1*	*(informations d'authentification par défaut du système)* dsp *sur:* chg	crt
Créer un canal	*(canal par défaut du système)* dsp	crt
Créer un canal (remplacer) *voir la remarque 1*	*(canal par défaut du système)* dsp *vers:* chg	crt
Créer un programme d'écoute de canal	*(programme d'écoute par défaut du système)* dsp	crt
Créer un programme d'écoute de canal (remplacement) *voir la remarque 1*	*(programme d'écoute par défaut du système)* dsp *sur:* chg	crt
Créer un canal de connexion client	*(canal par défaut du système)* dsp	crt
Créer un canal de connexion client (remplacement) *voir la remarque 1*	*(canal par défaut du système)* dsp *vers:* chg	crt
Créer une liste de noms	*(liste de noms par défaut du système)* dsp	crt
Créer une liste de noms (remplacer) *voir la remarque 1*	*(liste de noms par défaut du système)* dsp *sur:* dsp et chg	crt
Créer un processus	*(processus par défaut du système)* dsp	crt
Créer un processus (remplacer) *voir la remarque 1*	*(processus par défaut du système)* dsp *sur:* chg	crt
Créer une file d'attente	*(file d'attente par défaut du système)* dsp	crt
Créer une file d'attente (remplacement) *voir la remarque 1*	*(file d'attente par défaut du système)* dsp *vers:* dsp et chg	crt
Création de service	*(file d'attente par défaut du système)* dsp	crt
Create Service (Replace) *voir la remarque 1*	*(file d'attente par défaut du système)* dsp *vers:* chg	crt
Supprimer des informations d'authentification	dsp et dlt	Non applicable

Tableau 1. Windows, HP Integrity NonStop Server, systèmes UNIX and Linux -droits sur les objets (suite)		
Commande	**Droits sur les objets WebSphere MQ**	**Droits sur les classes (pour le type d'objet)**
Supprimer l'enregistrement de droits d'accès	*(objet gestionnaire de files d'attente)* chg *voir la remarque 4*	*voir la remarque 4*
Supprimer un canal	dsp et dlt	Non applicable
Supprimer le programme d'écoute de canal	dsp et dlt	Non applicable
Supprimer le canal de connexion client	dsp et dlt	Non applicable
Supprimer une liste de noms	dsp et dlt	Non applicable
Supprimer un processus	dsp et dlt	Non applicable
Supprimer une file d'attente	dsp et dlt	Non applicable
Supprimer le service	dsp et dlt	Non applicable
Consulter des informations d'authentification	dsp	Non applicable
Consulter des enregistrements de droits	*voir la remarque 4*	*voir la remarque 4*
Consulter un canal	dsp	Non applicable
Consulter le programme d'écoute de canal	dsp	Non applicable
Interroger le statut du canal (pour **ChannelType** MQCHT_CLSSDR)	inq	Non applicable
Consulter le canal de connexion client	dsp	Non applicable
Consulter une liste de noms	dsp	Non applicable
Consulter un processus	dsp	Non applicable
Consulter la file d'attente	dsp	Non applicable
Consulter les gestionnaires de files d'attente	*voir la remarque 3*	Non applicable
Consulter le statut d'une file d'attente	dsp	Non applicable
Consulter un service	dsp	Non applicable
Envoyer une commande PING à un canal	ctrl	Non applicable
Envoyer une commande Ping à un gestionnaire de files d'attente	*voir la remarque 3*	Non applicable
Régénérer un gestionnaire de files d'attente	(objet gestionnaire de files d'attente) chg	Non applicable

Tableau 1. Windows, HP Integrity NonStop Server, systèmes UNIX and Linux -droits sur les objets (suite)		
Commande	**Droits sur les objets WebSphere MQ**	**Droits sur les classes (pour le type d'objet)**
Actualiser la sécurité (pour **SecurityType** MQSECTYPE_SSL)	(objet gestionnaire de files d'attente) chg	Non applicable
Réinitialisation du canal	ctrlx	Non applicable
Réinitialiser un gestionnaire de files d'attente	(objet gestionnaire de files d'attente) chg	Non applicable
Réinitialiser les statistiques de file d'attente	dsp et chg	Non applicable
Résolution du canal	ctrlx	Non applicable
Définir l'enregistrement de droits d'accès	*(objet gestionnaire de files d'attente)* chg *voir la remarque 4*	*voir la remarque 4*
Démarrer un canal	ctrl	Non applicable
Arrêter le canal	ctrl	Non applicable
Arrêter une connexion	(objet gestionnaire de files d'attente) chg	Non applicable
Démarrage du programme d'écoute	ctrl	Non applicable
Arrêt du programme d'écoute	ctrl	Non applicable
Démarrer le service	ctrl	Non applicable
Arrêter le service	ctrl	Non applicable
Echap	*Voir la remarque 2*	*Voir la remarque 2*

Remarques :

1. Cette commande s'applique si l'objet à remplacer existe, sinon la vérification des droits est celle de la création ou de la copie sans remplacement.
2. Les droits requis sont déterminés par la commande MQSC définie par le texte d'échappement et sont équivalents à l'une des commandes précédentes.
3. Pour traiter une commande PCF, l'ID utilisateur doit disposer des droits dsp sur l'objet gestionnaire de files d'attente sur le système cible.
4. Cette commande PCF est autorisée sauf si le serveur de commandes a été démarré avec le paramètre -a. Par défaut, le serveur de commandes démarre lorsque le gestionnaire de files d'attente est démarré, sans le paramètre -a. Pour plus d'informations, voir le guide d'administration du système.
5. L'octroi à un ID utilisateur de droits d'accès *chg* pour un gestionnaire de files d'attente permet de définir des enregistrements de droits d'accès pour tous les groupes et utilisateurs. N'accordez pas ce droit à des utilisateurs ou à des applications ordinaires.

WebSphere MQ fournit également des points d'exit de sécurité de canal afin que vous puissiez fournir vos propres programmes d'exit utilisateur pour le contrôle de la sécurité. Des détails sont fournis dans le manuel Affichage d'un canal .

Utilisation de MQAI pour simplifier l'utilisation des fichiers PCF

MQAI est une interface d'administration de WebSphere MQ qui est disponible sur les plateformes AIX, HP-UX, IBM i, Linux, Solaris et Windows .

MQAI effectue des tâches d'administration sur un gestionnaire de files d'attente en utilisant des *sacs de données*. Les sacs de données vous permettent de gérer les propriétés (ou paramètres) des objets d'une manière plus simple que l'utilisation de fichiers PCF.

Utilisez MQAI de l'une des manières suivantes:

Pour simplifier l'utilisation des messages PCF
L'interface MQAI permet d'administrer facilement WebSphere MQ; il n'est pas nécessaire d'écrire vos propres messages PCF, ce qui permet d'éviter les problèmes liés aux structures de données complexes.

Pour transmettre des paramètres dans des programmes écrits à l'aide d'appels MQI, le message PCF doit contenir la commande et les détails de la chaîne ou des données entières. Pour ce faire, vous avez besoin de plusieurs instructions dans votre programme pour chaque structure et de l'espace mémoire doit être alloué. Cette tâche peut être longue et laborieuse.

Les programmes écrits à l'aide des MQAI transmettent les paramètres dans le sac de données approprié et vous n'avez besoin que d'une seule instruction pour chaque structure. L'utilisation de sacs de données MQAI vous évite de devoir gérer des grappes et d'allouer du stockage, et fournit un certain degré d'isolement par rapport aux détails du PCF.

Pour gérer les conditions d'erreur plus facilement
Il est difficile d'obtenir des codes retour à partir des commandes PCF, mais l'interface MQAI permet au programme de gérer plus facilement les conditions d'erreur.

Une fois que vous avez créé et rempli votre sac de données, vous pouvez envoyer un message de commande d'administration au serveur de commandes d'un gestionnaire de files d'attente, à l'aide de l'appel `mqExecute` , qui attend les messages de réponse. L'appel `mqExecute` gère l'échange avec le serveur de commandes et renvoie les réponses dans un *sac de réponse*.

Pour plus d'informations sur MQAI, voir «Introduction à l'interface d'administration IBM WebSphere MQ (MQAI)», à la page 21.

Introduction à l'interface d'administration IBM WebSphere MQ (MQAI)

L'interface d'administration IBM WebSphere MQ (MQAI) est une interface de programmation d' IBM WebSphere MQ. Il effectue des tâches d'administration sur un gestionnaire de files d'attente IBM WebSphere MQ à l'aide de sacs de données pour gérer les propriétés (ou les paramètres) des objets d'une manière plus simple que l'utilisation des formats PCF (Programmable Command Formats).

Concepts et terminologie MQAI

MQAI est une interface de programmation de WebSphere MQ, qui utilise le langage C et Visual Basic pour Windows. Il est disponible sur les plateformes autres que z/OS.

Il effectue des tâches d'administration sur un gestionnaire de files d'attente WebSphere MQ à l'aide de sacs de données. Les sacs de données vous permettent de gérer les propriétés (ou paramètres) des objets d'une manière plus simple que l'utilisation de l'autre interface d'administration, les PCF (Programmable Command Formats). L'interface MQAI offre une manipulation plus facile des fichiers PCF que l'utilisation des appels MQGET et MQPUT.

Pour plus d'informations sur les sacs de données, voir «Sacs de données», à la page 48. Pour plus d'informations sur les fichiers PCF, voir «Présentation des commandes PCF (Programmable Command Formats)», à la page 9

Utilisation de MQAI

Vous pouvez utiliser l'interface MQAI pour:

- Simplifiez l'utilisation des messages PCF. L'interface MQAI est un moyen simple d'administrer WebSphere MQ; vous n'avez pas à écrire vos propres messages PCF et vous évitez ainsi les problèmes associés à des structures de données complexes.
- Gérez les conditions d'erreur plus facilement. Il est difficile d'obtenir des codes retour à partir des commandes du script WebSphere MQ (MQSC), mais l'interface MQAI facilite la gestion des conditions d'erreur par le programme.
- Echange de données entre les applications. Les données d'application sont envoyées au format PCF et compressées et décompressées par MQAI. Si vos données de message se composent d'entiers et de chaînes de caractères, vous pouvez utiliser l'interface MQAI pour tirer parti de la conversion de données intégrée WebSphere MQ pour les données PCF. Cela évite d'avoir à écrire des exits de conversion de données. Pour plus d'informations sur l'utilisation de MQAI pour administrer WebSphere MQ et pour échanger des données entre des applications, voir «Utilisation de MQAI pour simplifier l'utilisation des fichiers PCF», à la page 20.

Exemples d'utilisation de MQAI

La liste ci-dessous fournit des exemples de programmes illustrant l'utilisation de MQAI. Les exemples effectuent les tâches suivantes:

1. Créez une file d'attente locale. «Exemple de programme C pour la création d'une file d'attente locale (amqsaicq.c)», à la page 23
2. Affichez les événements à l'écran à l'aide d'un moniteur d'événements simple. «Exemple de programme C pour l'affichage d'événements à l'aide d'un moniteur d'événements (amqsaiem.c)», à la page 26
3. Imprimez la liste de toutes les files d'attente locales et de leur profondeur actuelle. «Exemple de programme C pour l'interrogation des files d'attente et des informations d'impression (amqsailq.c)», à la page 38
4. Imprimez la liste de tous les canaux et de leurs types. «Exemple de programme C pour l'interrogation d'objets de canal (amqsaicl.c)», à la page 33

Génération de votre application MQAI

Pour générer votre application à l'aide de MQAI, vous établissez un lien vers les mêmes bibliothèques que pour WebSphere MQ. Pour plus d'informations sur la génération de vos applications WebSphere MQ , voir Génération d'une application WebSphere MQ .

Conseils et astuces pour la configuration de WebSphere MQ à l'aide de MQAI

MQAI utilise des messages PCF pour envoyer des commandes d'administration au serveur de commandes au lieu de traiter directement avec le serveur de commandes lui-même. Vous trouverez des conseils pour la configuration de WebSphere MQ à l'aide de MQAI dans «Conseils et astuces pour la configuration de IBM WebSphere MQ», à la page 42

Interface d'administration IBM WebSphere MQ (MQAI)

IBM WebSphere MQ for Windows, AIX, Linux, HP-UXet Solaris prennent en charge l'interface d'administration IBM WebSphere MQ (MQAI). L'interface MQAI est une interface de programmation d' IBM WebSphere MQ qui offre une alternative à l'interface MQI pour l'envoi et la réception de fichiers PCF.

L'interface MQAI utilise des *sacs de données* qui vous permettent de gérer les propriétés (ou paramètres) des objets plus facilement que l'utilisation de fichiers PCF directement via l'interface MQAI.

L'interface MQAI facilite l'accès à la programmation des messages PCF en transmettant des paramètres dans le sac de données, de sorte qu'une seule instruction est requise pour chaque structure. Cet accès élimine le besoin pour le programmeur de gérer les grappes et d'allouer de la mémoire, et fournit un certain isolement des détails de PCF.

L'interface MQAI administre WebSphere MQ en envoyant des messages PCF au serveur de commandes et en attendant une réponse.

L'interface MQAI est décrite dans la deuxième section de ce manuel. Voir la documentation Utilisation de Java pour une description d'une interface de modèle d'objet de composant pour MQAI.

Exemple de programme C pour la création d'une file d'attente locale (amqsaicq.c)

L'exemple de programme C `amqsaicq.c` crée une file d'attente locale à l'aide de MQAI.

```
/**********************************************************************/
/*                                                                    */
/* Program name: AMQSAICQ.C                                           */
/*                                                                    */
/* Description:  Sample C program to create a local queue using the   */
/*               WebSphere MQ Administration Interface (MQAI).        */
/*                                                                    */
/* Statement:    Licensed Materials - Property of IBM                 */
/*                                                                    */
/*               84H2000, 5765-B73                                    */
/*               84H2001, 5639-B42                                    */
/*               84H2002, 5765-B74                                    */
/*               84H2003, 5765-B75                                    */
/*               84H2004, 5639-B43                                    */
/*                                                                    */
/*               (C) Copyright IBM Corp. 1999, 2024.                  */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Function:                                                          */
/*    AMQSAICQ is a sample C program that creates a local queue and is an */
/*    example of the use of the mqExecute call.                       */
/*                                                                    */
/*     - The name of the queue to be created is a parameter to the program. */
/*                                                                    */
/*     - A PCF command is built by placing items into an MQAI bag.    */
/*       These are:-                                                  */
/*            - The name of the queue                                 */
/*            - The type of queue required, which, in this case, is local. */
/*                                                                    */
/*     - The mqExecute call is executed with the command MQCMD_CREATE_Q. */
/*       The call generates the correct PCF structure.                */
/*       The call receives the reply from the command server and formats into */
/*       the response bag.                                            */
/*                                                                    */
/*     - The completion code from the mqExecute call is checked and if there */
/*       is a failure from the command server then the code returned by the */
/*       command server is retrieved from the system bag that is      */
/*       embedded in the response bag to the mqExecute call.          */
/*                                                                    */
/* Note: The command server must be running.                          */
/*                                                                    */
/*                                                                    */

/**********************************************************************/
/*                                                                    */
/* AMQSAICQ has 2 parameters - the name of the local queue to be created */
/*                           - the queue manager name (optional)      */
/*                                                                    */
/**********************************************************************/
/**********************************************************************/
/* Includes                                                           */
/**********************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                        */
#include <cmqcfc.h>                     /* PCF                        */
#include <cmqbc.h>                      /* MQAI                       */

void CheckCallResult(MQCHAR *, MQLONG , MQLONG );
void CreateLocalQueue(MQHCONN, MQCHAR *);

int main(int argc, char *argv[])
```

```
{
   MQHCONN hConn;                          /* handle to WebSphere MQ connection   */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name   */
   MQLONG connReason;                      /* MQCONN reason code              */
   MQLONG compCode;                        /* completion code                 */
   MQLONG reason;                          /* reason code                     */

   /*************************************************************************/
   /* First check the required parameters                                   */
   /*************************************************************************/
   printf("Sample Program to Create a Local Queue\n");
   if (argc < 2)
   {
     printf("Required parameter missing - local queue name\n");
     exit(99);
   }

   /*************************************************************************/
   /* Connect to the queue manager                                          */
   /*************************************************************************/
   if (argc > 2)
      strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
      MQCONN(QMName, &hConn, &compCode, &connReason);

/*************************************************************************/
/* Report reason and stop if connection failed                           */
/*************************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("MQCONN", compCode, connReason);
      exit( (int)connReason);
   }


/*************************************************************************/
/* Call the routine to create a local queue, passing the handle to the   */
/* queue manager and also passing the name of the queue to be created.   */
/*************************************************************************/
   CreateLocalQueue(hConn, argv[1]);

   /*************************************************************************/
   /* Disconnect from the queue manager if not already connected            */
   /*************************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
   }
   return 0;

}
/*************************************************************************/
/*                                                                       */
/* Function:    CreateLocalQueue                                         */
/* Description: Create a local queue by sending a PCF command to the command  */
/*              server.                                                  */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/* Input Parameters:  Handle to the queue manager                        */
/*                    Name of the queue to be created                    */
/*                                                                       */
/* Output Parameters: None                                               */
/*                                                                       */
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.     */
/*        The call generates the correct PCF structure.                  */
/*        The default options to the call are used so that the command is sent*/
/*        to the SYSTEM.ADMIN.COMMAND.QUEUE.                             */
/*        The reply from the command server is placed on a temporary dynamic  */
/*        queue.                                                         */
/*        The reply is read from the temporary queue and formatted into the   */
/*        response bag.                                                  */
/*                                                                       */
/*        The completion code from the mqExecute call is checked and if there */
/*        is a failure from the command server then the code returned by the  */
/*        command server is retrieved from the system bag that is        */
/*        embedded in the response bag to the mqExecute call.            */
/*                                                                       */
/*************************************************************************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG reason;                          /* reason code                     */
```

```
MQLONG compCode;                        /* completion code                */
MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute       */
MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
MQHBAG resultBag;                       /* result bag from mqExecute       */
MQLONG mqExecuteCC;                     /* mqExecute completion code       */
MQLONG mqExecuteRC;                     /* mqExecute reason code           */

printf("\nCreating Local Queue %s\n\n", qName);


/*************************************************************************/
/* Create a command Bag for the mqExecute call. Exit the function if the   */
/* create fails.                                                         */
/*************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
CheckCallResult("Create the command bag", compCode, reason);
if (compCode !=MQCC_OK)
   return;

/*************************************************************************/
/* Create a response Bag for the mqExecute call, exit the function if the  */
/* create fails.                                                         */
/*************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
CheckCallResult("Create the response bag", compCode, reason);
if (compCode !=MQCC_OK)
   return;

/*************************************************************************/
/* Put the name of the queue to be created into the command bag. This will */
/* be used by the mqExecute call.                                        */
/*************************************************************************/
mqAddString(commandBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, qName, &compCode,
            &reason);
CheckCallResult("Add q name to command bag", compCode, reason);

/*************************************************************************/
/* Put queue type of local into the command bag. This will be used by the  */
/* mqExecute call.                                                       */
/*************************************************************************/
mqAddInteger(commandBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type to command bag", compCode, reason);

/*************************************************************************/
/* Send the command to create the required local queue.                  */
/* The mqExecute call will create the PCF structure required, send it to   */
/* the command server and receive the reply from the command server into   */
/* the response bag.                                                     */
/*************************************************************************/
mqExecute(hConn,                    /* WebSphere MQ connection handle           */
          MQCMD_CREATE_Q,           /* Command to be executed                */
          MQHB_NONE,                /* No options bag                        */
          commandBag,               /* Handle to bag containing commands     */
          responseBag,              /* Handle to bag to receive the response*/
          MQHO_NONE,                /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,                /* Create a dynamic q for the response   */
          &compCode,                /* Completion code from the mqExecute    */
          &reason);                 /* Reason code from mqExecute call       */

if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName>\n")
   MQDISC(&hConn, &compCode, &reason);
   CheckCallResult("MQDISC", compCode, reason);
   exit(98);
}


/*************************************************************************/
/* Check the result from mqExecute call and find the error if it failed.   */
/*************************************************************************/
if ( compCode == MQCC_OK )
   printf("Local queue %s successfully created\n", qName);
else
{
   printf("Creation of local queue %s failed: Completion Code = %d
            qName, compCode, reason);
   if (reason == MQRCCF_COMMAND_FAILED)
   {
      /*********************************************************************/
      /* Get the system bag handle out of the mqExecute response bag.      */
      /* This bag contains the reason from the command server why the      */
```

```
         /* command failed.                                                */
         /*****************************************************************/
         mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
                      &reason);
         CheckCallResult("Get the result bag handle", compCode, reason);

         /*****************************************************************/
         /* Get the completion code and reason code, returned by the command  */
         /* server, from the embedded error bag.                          */
         /*****************************************************************/
         mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                          &compCode, &reason);
         CheckCallResult("Get the completion code from the result bag",
                          compCode, reason);
         mqInquireInteger(resultBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                          &compCode, &reason);
         CheckCallResult("Get the reason code from the result bag", compCode,
                          reason);
         printf("Error returned by the command server: Completion code = %d :
                 Reason = %d\n", mqExecuteCC, mqExecuteRC);
      }
   }
   /*****************************************************************************/
   /* Delete the command bag if successfully created.                           */
   /*****************************************************************************/
   if (commandBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&commandBag, &compCode, &reason);
      CheckCallResult("Delete the command bag", compCode, reason);
   }

   /*****************************************************************************/
   /* Delete the response bag if successfully created.                          */
   /*****************************************************************************/
   if (responseBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&responseBag, &compCode, &reason);
      CheckCallResult("Delete the response bag", compCode, reason);
   }
} /* end of CreateLocalQueue */


/*****************************************************************************/
/*                                                                           */
/* Function: CheckCallResult                                                 */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Description of call                                    */
/*                    Completion code                                        */
/*                    Reason code                                            */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: Display the description of the call, the completion code and the   */
/*        reason code if the completion code is not successful               */
/*                                                                           */
/*****************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %d :
                 Reason = %d\n", callText, cc, rc);

}
```

Exemple de programme C pour l'affichage d'événements à l'aide d'un moniteur d'événements (amqsaiem.c)

L'exemple de programme C `amqsaiem.c` illustre un moniteur d'événements de base à l'aide de MQAI.

```
*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAIEM.C                                                  */
/*                                                                           */
/* Description:  Sample C program to demonstrate a basic event monitor       */
/*               using the WebSphere MQ Admin Interface (MQAI).              */
```

```
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 1999, 2024. All Rights Reserved.                  */
/*                                                                           */
/* US Government Users Restricted Rights - Use, duplication or               */
/* disclosure restricted by GSA ADP Schedule Contract with                   */
/* IBM Corp.                                                                 */
/*****************************************************************************/
/*                                                                           */
/* Function:                                                                 */
/*    AMQSAIEM is a sample C program that demonstrates how to write a simple  */
/*    event monitor using the mqGetBag call and other MQAI calls.            */
/*                                                                           */
/*    The name of the event queue to be monitored is passed as a parameter    */
/*    to the program. This would usually be one of the system event queues:-  */
/*            SYSTEM.ADMIN.QMGR.EVENT        Queue Manager events            */
/*            SYSTEM.ADMIN.PERFM.EVENT       Performance events              */
/*            SYSTEM.ADMIN.CHANNEL.EVENT     Channel events                  */
/*            SYSTEM.ADMIN.LOGGER.EVENT      Logger events                   */
/*                                                                           */
/*    To monitor the queue manager event queue or the performance event queue,*/
/*    the attributes of the queue manager needs to be changed to enable       */
/*    these events. For more information about this, see Part 1 of the        */
/*    Programmable System Management book. The queue manager attributes can   */
/*    be changed using either MQSC commands or the MQAI interface.            */
/*    Channel events are enabled by default.                                  */
/*                                                                           */
/* Program logic                                                             */
/*    Connect to the Queue Manager.                                          */
/*    Open the requested event queue with a wait interval of 30 seconds.      */
/*    Wait for a message, and when it arrives get the message from the queue  */
/*    and format it into an MQAI bag using the mqGetBag call.                */
/*    There are many types of event messages and it is beyond the scope of    */
/*    this sample to program for all event messages. Instead the program      */
/*    prints out the contents of the formatted bag.                          */
/*    Loop around to wait for another message until either there is an error  */
/*    or the wait interval of 30 seconds is reached.                         */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored   */
/*                           - the queue manager name (optional)             */
/*                                                                           */
/*****************************************************************************

/*****************************************************************************/
/* Includes                                                                  */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                               */
#include <cmqcfc.h>                     /* PCF                               */
#include <cmqbc.h>                      /* MQAI                              */

/*****************************************************************************/
/* Macros                                                                    */
/*****************************************************************************/
#if MQAT_DEFAULT == MQAT_WINDOWS_NT
  #define Int64 "I64"
#elif defined(MQ_64_BIT)
  #define Int64 "l"
#else
  #define Int64 "ll"
#endif

/*****************************************************************************/
/* Function prototypes                                                       */
/*****************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);
void GetQEvents(MQHCONN, MQCHAR *);
int PrintBag(MQHBAG);
int PrintBagContents(MQHBAG, int);

/*****************************************************************************/
/* Function: main                                                            */
/*****************************************************************************/
int main(int argc, char *argv[])
{
```

```
   MQHCONN hConn;                          /* handle to connection            */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QM name               */
   MQLONG reason;                          /* reason code                     */
   MQLONG connReason;                      /* MQCONN reason code              */
   MQLONG compCode;                        /* completion code                 */

   /*******************************************************************/
   /* First check the required parameters                             */
   /*******************************************************************/
   printf("Sample Event Monitor (times out after 30 secs)\n");
   if (argc < 2)
   {
     printf("Required parameter missing - event queue to be monitored\n");
     exit(99);
   }

   /*******************************************************************/
   /* Connect to the queue manager                                    */
   /*******************************************************************/
   if (argc > 2)
     strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(QMName, &hConn, &compCode, &connReason);
   /*******************************************************************/
   /* Report the reason and stop if the connection failed             */
   /*******************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("MQCONN", compCode, connReason);
      exit( (int)connReason);
   }

   /*******************************************************************/
   /* Call the routine to open the event queue and format any event messages  */
   /* read from the queue.                                            */
   /*******************************************************************/
   GetQEvents(hConn, argv[1]);

   /*******************************************************************/
   /* Disconnect from the queue manager if not already connected      */
   /*******************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
   }

   return 0;

}

/*******************************************************************/
/*                                                                 */
/* Function: CheckCallResult                                       */
/*                                                                 */
/*******************************************************************/
/*                                                                 */
/* Input Parameters:  Description of call                          */
/*                    Completion code                              */
/*                    Reason code                                  */
/*                                                                 */
/* Output Parameters: None                                         */
/*                                                                 */
/* Logic: Display the description of the call, the completion code and the    */
/*        reason code if the completion code is not successful     */
/*                                                                 */
/*******************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %d : Reason = %d\n",
                 callText, cc, rc);

}

/*******************************************************************/
/*                                                                 */
/* Function: GetQEvents                                            */
/*                                                                 */
/*******************************************************************/
/*                                                                 */
/* Input Parameters:  Handle to the queue manager                  */
/*                    Name of the event queue to be monitored      */
```

```
/*                                                                  */
/* Output Parameters: None                                          */
/*                                                                  */
/* Logic:   Open the event queue.                                   */
/*          Get a message off the event queue and format the message into */
/*          a bag.                                                  */
/*          A real event monitor would need to be programmed to deal with */
/*          each type of event that it receives from the queue. This is */
/*          outside the scope of this sample, so instead, the contents of */
/*          the bag are printed.                                    */
/*          The program waits for 30 seconds for an event message and then */
/*          terminates if no more messages are available.           */
/*                                                                  */
/********************************************************************/
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG openReason;                      /* MQOPEN reason code        */
   MQLONG reason;                          /* reason code               */
   MQLONG compCode;                        /* completion code           */
   MQHOBJ eventQueue;                      /* handle to event queue     */

   MQHBAG eventBag = MQHB_UNUSABLE_HBAG;   /* event bag to receive event msg */
   MQOD   od = {MQOD_DEFAULT};             /* Object Descriptor         */
   MQMD   md = {MQMD_DEFAULT};             /* Message Descriptor        */
   MQGMO  gmo = {MQGMO_DEFAULT};           /* get message options       */
   MQLONG bQueueOK = 1;                    /* keep reading msgs while true */

   /*****************************************************************/
   /* Create an Event Bag in which to receive the event.            */
   /* Exit the function if the create fails.                        */
   /*****************************************************************/
   mqCreateBag(MQCBO_USER_BAG, &eventBag, &compCode, &reason);
   CheckCallResult("Create event bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /*****************************************************************/
   /* Open the event queue chosen by the user                       */
   /*****************************************************************/
   strncpy(od.ObjectName, qName, (size_t)MQ_Q_NAME_LENGTH);
   MQOPEN(hConn, &od, MQOO_INPUT_AS_Q_DEF+MQOO_FAIL_IF_QUIESCING, &eventQueue,
          &compCode, &openReason);
   CheckCallResult("Open event queue", compCode, openReason);

   /*****************************************************************/
   /* Set the GMO options to control the action of the get message from the */
   /* queue.                                                        */
   /*****************************************************************/
   gmo.WaitInterval = 30000;               /* 30 second wait for message */
   gmo.Options = MQGMO_WAIT + MQGMO_FAIL_IF_QUIESCING + MQGMO_CONVERT;
   gmo.Version = MQGMO_VERSION_2;          /* Avoid need to reset Message ID */
   gmo.MatchOptions = MQMO_NONE;           /* and Correlation ID after every */
                                /* mqGetBag
   /*****************************************************************/
   /* If open fails, we cannot access the queue and must stop the monitor. */
   /*****************************************************************/
   if (compCode != MQCC_OK)
      bQueueOK = 0;

   /*****************************************************************/
   /* Main loop to get an event message when it arrives             */
   /*****************************************************************/
   while (bQueueOK)
   {
     printf("\nWaiting for an event\n");

     /****************************************************************/
     /* Get the message from the event queue and convert it into the event */
     /* bag.                                                         */
     /****************************************************************/
     mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

     /****************************************************************/
     /* If get fails, we cannot access the queue and must stop the monitor. */
     /****************************************************************/
     if (compCode != MQCC_OK)
     {
        bQueueOK = 0;

        /*************************************************************/
        /* If get fails because no message available then we have timed out, */
        /* so report this, otherwise report an error.                */
```

```
        /*****************************************************************/
        if (reason == MQRC_NO_MSG_AVAILABLE)
        {
           printf("No more messages\n");
        }
        else
        {
           CheckCallResult("Get bag", compCode, reason);
        }
     }

     /*****************************************************************/
     /* Event message read - Print the contents of the event bag      */
     /*****************************************************************/
     else
     {
       if ( PrintBag(eventBag) )
           printf("\nError found while printing bag contents\n");

     }  /* end of msg found */
   } /* end of main loop */
   /*****************************************************************/
   /* Close the event queue if successfully opened                  */
   /*****************************************************************/
   if (openReason == MQRC_NONE)
   {
      MQCLOSE(hConn, &eventQueue, MQCO_NONE, &compCode, &reason);
      CheckCallResult("Close event queue", compCode, reason);
   }

   /*****************************************************************/
   /* Delete the event bag if successfully created.                 */
   /*****************************************************************/
   if (eventBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&eventBag, &compCode, &reason);
      CheckCallResult("Delete the event bag", compCode, reason);
   }

} /* end of GetQEvents */

/*****************************************************************************/
/*                                                                           */
/* Function: PrintBag                                                        */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Bag Handle                                             */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Returns:           Number of errors found                                 */
/*                                                                           */
/* Logic: Calls PrintBagContents to display the contents of the bag.         */
/*                                                                           */
/*****************************************************************************

int PrintBag(MQHBAG dataBag)
{
    int errors;

    printf("\n");
    errors = PrintBagContents(dataBag, 0);
    printf("\n");

    return errors;
}

/*****************************************************************************/
/*                                                                           */
/* Function: PrintBagContents                                                */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Bag Handle                                             */
/*                    Indentation level of bag                               */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Returns:           Number of errors found                                 */
/*                                                                           */
/* Logic: Count the number of items in the bag                               */
```

```
/*        Obtain selector and item type for each item in the bag.             */
/*        Obtain the value of the item depending on item type and display the */
/*        index of the item, the selector and the value.                      */
/*        If the item is an embedded bag handle then call this function again */
/*        to print the contents of the embedded bag increasing the            */
/*        indentation level.                                                  */
/*                                                                            */
/******************************************************************************/
int PrintBagContents(MQHBAG dataBag, int indent)
{
   /*****************************************************************************/
   /* Definitions                                                               */
   /*****************************************************************************/
   #define LENGTH 500                      /* Max length of string to be read*/
   #define INDENT 4                        /* Number of spaces to indent     */
                                           /* embedded bag display           */


   /*****************************************************************************/
   /* Variables                                                                 */
   /*****************************************************************************/
   MQLONG  itemCount;                      /* Number of items in the bag     */
   MQLONG  itemType;                       /* Type of the item               */
   int     i;                              /* Index of item in the bag       */
   MQCHAR  stringVal[LENGTH+1];            /* Value if item is a string      */
   MQBYTE  byteStringVal[LENGTH];          /* Value if item is a byte string */
   MQLONG  stringLength;                   /* Length of string value         */
   MQLONG  ccsid;                          /* CCSID of string value          */
   MQINT32 iValue;                         /* Value if item is an integer    */
   MQINT64 i64Value;                       /* Value if item is a 64-bit      */
                                           /* integer                        */
   MQLONG  selector;                       /* Selector of item               */
   MQHBAG  bagHandle;                      /* Value if item is a bag handle  */
   MQLONG  reason;                         /* reason code                    */
   MQLONG  compCode;                       /* completion code                */
   MQLONG  trimLength;                     /* Length of string to be trimmed */
   int     errors = 0;                     /* Count of errors found          */
   char    blanks[] = "                      "; /* Blank string used to      */
                                                /* indent display            */

   /*****************************************************************************/
   /* Count the number of items in the bag                                      */
   /*****************************************************************************/
   mqCountItems(dataBag, MQSEL_ALL_SELECTORS, &itemCount, &compCode, &reason);

   if (compCode != MQCC_OK)
      errors++;
   else
   {
      printf("
      printf("
      printf("
   }

   /*****************************************************************************/
   /* If no errors found, display each item in the bag                          */
   /*****************************************************************************/
   if (!errors)
   {
      for (i = 0; i < itemCount; i++)
      {

          /*************************************************************/
          /* First inquire the type of the item for each item in the bag      */
          /*************************************************************/
          mqInquireItemInfo(dataBag,             /* Bag handle             */
                            MQSEL_ANY_SELECTOR,  /* Item can have any selector*/
                            i,                   /* Index position in the bag */
                            &selector,           /* Actual value of selector  */
                                                 /* returned by call          */
                            &itemType,           /* Actual type of item       */
                                                 /* returned by call          */
                            &compCode,           /* Completion code           */
                            &reason);            /* Reason Code               */

          if (compCode != MQCC_OK)
             errors++;

          switch(itemType)
          {
          case MQITEM_INTEGER:
               /*************************************************************/
```

```
      /* Item is an integer. Find its value and display its index,   */
      /* selector and value.                                         */
      /*************************************************************/
      mqInquireInteger(dataBag,         /* Bag handle                */
                       MQSEL_ANY_SELECTOR, /* Allow any selector     */
                       i,               /* Index position in the bag */
                       &iValue,         /* Returned integer value
                       &compCode,       /* Completion code           */
                       &reason);        /* Reason Code               */

      if (compCode != MQCC_OK)
         errors++;
      else
         printf("%.*s  %-2d     %-4d     (%d)\n",
                 indent, blanks, i, selector, iValue);
      break

   case MQITEM_INTEGER64:
      /*************************************************************/
      /* Item is a 64-bit integer. Find its value and display its    */
      /* index, selector and value.                                  */
      /*************************************************************/
      mqInquireInteger64(dataBag,       /* Bag handle                */
                         MQSEL_ANY_SELECTOR, /* Allow any selector   */
                         i,             /* Index position in the bag */
                         &i64Value,     /* Returned integer value    */
                         &compCode,     /* Completion code           */
                         &reason);      /* Reason Code               */

      if (compCode != MQCC_OK)
         errors++;
      else
         printf("%.*s  %-2d     %-4d     (%"Int64"d)\n",
                 indent, blanks, i, selector, i64Value);
      break;


   case MQITEM_STRING:
      /*************************************************************/
      /* Item is a string. Obtain the string in a buffer, prepare    */
      /* the string for displaying and display the index, selector,  */
      /* string and Character Set ID.                                */
      /*************************************************************/
      mqInquireString(dataBag,          /* Bag handle                */
                      MQSEL_ANY_SELECTOR, /* Allow any selector      */
                      i,                /* Index position in the bag */
                      LENGTH,           /* Maximum length of buffer  */
                      stringVal,        /* Buffer to receive string  */
                      &stringLength,    /* Actual length of string   */
                      &ccsid,           /* Coded character set id    */
                      &compCode,        /* Completion code           */
                      &reason);         /* Reason Code               */

      /*************************************************************/
      /* The call can return a warning if the string is too long for */
      /* the output buffer and has been truncated, so only check     */
      /* explicitly for call failure.                                */
      /*************************************************************/
      if (compCode == MQCC_FAILED)
          errors++;
      else
      {
         /*********************************************************/
         /* Remove trailing blanks from the string and terminate with*/
         /* a null. First check that the string should not have been */
         /* longer than the maximum buffer size allowed.            */
         /*********************************************************/
         if (stringLength > LENGTH)
            trimLength = LENGTH;
         else
            trimLength = stringLength;
         mqTrim(trimLength, stringVal, &compCode, &reason);
         printf("%.*s  %-2d     %-4d     '%s' %d\n",
                 indent, blanks, i, selector, stringVal, ccsid);
      }
      break;

   case MQITEM_BYTE_STRING:
      /*************************************************************/
      /* Item is a byte string. Obtain the byte string in a buffer,  */
      /* prepare the byte string for displaying and display the      */
      /* index, selector and string.                                 */
```

```
                /*************************************************************/
                mqInquireByteString(dataBag,      /* Bag handle               */
                                    MQSEL_ANY_SELECTOR, /* Allow any selector  */
                                    i,            /* Index position in the bag */
                                    LENGTH,       /* Maximum length of buffer  */
                                    byteStringVal, /* Buffer to receive string */
                                    &stringLength, /* Actual length of string  */
                                    &compCode,    /* Completion code           */
                                    &reason);     /* Reason Code

                /*************************************************************/
                /* The call can return a warning if the string is too long for */
                /* the output buffer and has been truncated, so only check     */
                /* explicitly for call failure.                                */
                /*************************************************************/
                if (compCode == MQCC_FAILED)
                    errors++;
                else
                {
                   printf("%.*s  %-2d     %-4d     X'",
                          indent, blanks, i, selector);

                   for (i = 0 ; i < stringLength ; i++)
                      printf("

                   printf("'\n");
                }
                break;

           case MQITEM_BAG:
                /*************************************************************/
                /* Item is an embedded bag handle, so call the PrintBagContents*/
                /* function again to display the contents.                     */
                /*************************************************************/
                mqInquireBag(dataBag,             /* Bag handle              */
                             MQSEL_ANY_SELECTOR,  /* Allow any selector      */
                             i,                   /* Index position in the bag */
                             &bagHandle,          /* Returned embedded bag hdle*/
                             &compCode,           /* Completion code         */
                             &reason);            /* Reason Code             */

                if (compCode != MQCC_OK)
                   errors++;
                else
                {
                   printf("%.*s  %-2d     %-4d     (%d)\n", indent, blanks, i,
                           selector, bagHandle);
                   if (selector == MQHA_BAG_HANDLE)
                      printf("
                   else
                      printf("
                   PrintBagContents(bagHandle, indent+INDENT);
                }
                break;

           default:
                printf("
                }
        }
     }
     return errors;
}
```

Exemple de programme C pour l'interrogation d'objets de canal (amqsaicl.c)

L'exemple de programme C `amqsaicl.c` demande des objets de canal à l'aide de MQAI.

```
 /*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAICL.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire channel objects                 */
/*               using the WebSphere MQ Administration Interface (MQAI)      */
/*                                                                           */
/* <N_OCO_COPYRIGHT>                                                         */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 2008, 2024. All Rights Reserved.                  */
```

```
/*                                                                  */
/* US Government Users Restricted Rights - Use, duplication or      */
/* disclosure restricted by GSA ADP Schedule Contract with          */
/* IBM Corp.                                                        */
/* <NOC_COPYRIGHT>                                                  */
/********************************************************************/
/*                                                                  */
/* Function:                                                        */
/*    AMQSAICL is a sample C program that demonstrates how to inquire */
/*    attributes of the local queue manager using the MQAI interface. In */
/*    particular, it inquires all channels and their types.         */
/*                                                                  */
/*     - A PCF command is built from items placed into an MQAI administration */
/*       bag.                                                       */
/*       These are:-                                                */
/*            - The generic channel name "*"                        */
/*            - The attributes to be inquired. In this sample we just want */
/*              name and type attributes                            */
/*                                                                  */
/*     - The mqExecute MQCMD_INQUIRE_CHANNEL call is executed.      */
/*       The call generates the correct PCF structure.              */
/*       The default options to the call are used so that the command is sent */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                         */
/*       The reply from the command server is placed on a temporary dynamic */
/*       queue.                                                     */
/*       The reply from the MQCMD_INQUIRE_CHANNEL is read from the  */
/*       temporary queue and formatted into the response bag.       */
/*                                                                  */
/*     - The completion code from the mqExecute call is checked and if there */
/*       is a failure from the command server, then the code returned by the */
/*       command server is retrieved from the system bag that has been */
/*       embedded in the response bag to the mqExecute call.        */
/*                                                                  */
/* Note: The command server must be running.                        */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* AMQSAICL has 2 parameter - the queue manager name (optional)     */
/*                          - output file (optional) default varies */
/********************************************************************/

/********************************************************************/
/* Includes                                                         */
/********************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#if (MQAT_DEFAULT == MQAT_OS400)
#include <recio.h>
#endif

#include <cmqc.h>                       /* MQI                      */
#include <cmqcfc.h>                     /* PCF                      */
#include <cmqbc.h>                      /* MQAI                     */
#include <cmqxc.h>                      /* MQCD                     */

/********************************************************************/
/* Function prototypes                                              */
/********************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/********************************************************************/
/* DataTypes                                                        */
/********************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
typedef _RFILE OUTFILEHDL;
#else
typedef FILE OUTFILEHDL;
#endif

/********************************************************************/
/* Constants                                                        */
/********************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "*SDR     ",    /* MQCHT_SENDER    */
  "*SVR     ",    /* MQCHT_SERVER    */
```

```
  "*RCVR    ",    /* MQCHT_RECEIVER  */
  "*RQSTR   ",    /* MQCHT_REQUESTER */
  "*ALL     ",    /* MQCHT_ALL       */
  "*CLTCN   ",    /* MQCHT_CLNTCONN  */
  "*SVRCONN ",    /* MQCHT_SVRCONN   */
  "*CLUSRCVR",    /* MQCHT_CLUSRCVR  */
  "*CLUSSDR "     /* MQCHT_CLUSSDR   */
};
#else
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "sdr      ",    /* MQCHT_SENDER    */
  "svr      ",    /* MQCHT_SERVER    */
  "rcvr     ",    /* MQCHT_RECEIVER  */
  "rqstr    ",    /* MQCHT_REQUESTER */
  "all      ",    /* MQCHT_ALL       */
  "cltconn  ",    /* MQCHT_CLNTCONN  */
  "svrcn    ",    /* MQCHT_SVRCONN   */
  "clusrcvr ",    /* MQCHT_CLUSRCVR  */
  "clussdr  "     /* MQCHT_CLUSSDR   */
};
#endif

/********************************************************************************/
/* Macros                                                                       */
/********************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
  #define OUTFILE "QTEMP/AMQSAICL(AMQSAICL)"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = _Ropen((fname),"wr, rtncode=Y");
  #define CLOSEOUTFILE(hdl) \
    _Rclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    _Rwrite((hdl),(buf),(buflen));

#elif (MQAT_DEFAULT == MQAT_UNIX)
  #define OUTFILE "/tmp/amqsaicl.txt"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#else
  #define OUTFILE "amqsaicl.txt"
  #define OPENOUTFILE(fname) \
    fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#endif

#define ChlType2String(t) ChlTypeMap[(t)-1].name

/********************************************************************************/
/* Function: main                                                               */
/********************************************************************************/
int main(int argc, char *argv[])
{
   /*****************************************************************************/
   /* MQAI variables                                                            */
   /*****************************************************************************/
   MQHCONN hConn;                          /* handle to MQ connection         */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name             */
   MQLONG reason;                          /* reason code                     */
   MQLONG connReason;                      /* MQCONN reason code              */
   MQLONG compCode;                        /* completion code                 */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute         */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
   MQHBAG cAttrsBag;                       /* bag containing chl attributes   */
   MQHBAG errorBag;                        /* bag containing cmd server error */
   MQLONG mqExecuteCC;                     /* mqExecute completion code       */
   MQLONG mqExecuteRC;                     /* mqExecute reason code           */
   MQLONG chlNameLength;                   /* Actual length of chl name       */
   MQLONG chlType;                         /* Channel type                    */
   MQLONG i;                               /* loop counter                    */
```

```
MQLONG numberOfBags;                    /* number of bags in response bag  */
MQCHAR chlName[MQ_OBJECT_NAME_LENGTH+1];/* name of chl extracted from bag  */
MQCHAR OutputBuffer[100];               /* output data buffer              */
OUTFILEHDL *outfp = NULL;               /* output file handle              */

/*********************************************************************/
/* Connect to the queue manager                                      */
/*********************************************************************/
if (argc &gt; 1)
   strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
MQCONN(qmName, &hConn;, &compCode;, &connReason;);

/*********************************************************************/
/* Report the reason and stop if the connection failed.              */
/*********************************************************************/
if (compCode == MQCC_FAILED)
{
   CheckCallResult("Queue Manager connection", compCode, connReason);
   exit( (int)connReason);
}

/*********************************************************************/
/* Open the output file                                              */
/*********************************************************************/
if (argc &gt; 2)
{
  OPENOUTFILE(outfp, argv[2]);
}
else
{
  OPENOUTFILE(outfp, OUTFILE);
}

if(outfp == NULL)
{
  printf("Could not open output file.\n");
  goto MOD_EXIT;
}
/*********************************************************************/
/* Create an admin bag for the mqExecute call                        */
/*********************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag;, &compCode;, &reason;);
CheckCallResult("Create admin bag", compCode, reason);

/*********************************************************************/
/* Create a response bag for the mqExecute call                      */
/*********************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag;, &compCode;, &reason;);
CheckCallResult("Create response bag", compCode, reason);

/*********************************************************************/
/* Put the generic channel name into the admin bag                   */
/*********************************************************************/
mqAddString(adminBag, MQCACH_CHANNEL_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode;, &reason;);
CheckCallResult("Add channel name", compCode, reason);

/*********************************************************************/
/* Put the channel type into the admin bag                           */
/*********************************************************************/
mqAddInteger(adminBag, MQIACH_CHANNEL_TYPE, MQCHT_ALL, &compCode;, &reason;);
CheckCallResult("Add channel type", compCode, reason);

/*********************************************************************/
/* Add an inquiry for various attributes                             */
/*********************************************************************/
mqAddInquiry(adminBag, MQIACH_CHANNEL_TYPE, &compCode;, &reason;);
CheckCallResult("Add inquiry", compCode, reason);

/*********************************************************************/
/* Send the command to find all the channel names and channel types.  */
/* The mqExecute call creates the PCF structure required, sends it to */
/* the command server, and receives the reply from the command server into */
/* the response bag. The attributes are contained in system bags that are  */
/* embedded in the response bag, one set of attributes per bag.       */
/*********************************************************************/
mqExecute(hConn,                    /* MQ connection handle            */
          MQCMD_INQUIRE_CHANNEL,    /* Command to be executed          */
          MQHB_NONE,                /* No options bag                  */
          adminBag,                 /* Handle to bag containing commands    */
          responseBag,              /* Handle to bag to receive the response*/
          MQHO_NONE,                /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
```

```
           MQHO_NONE,               /* Create a dynamic q for the response  */
           &compCode;,               /* Completion code from the mqexecute   */
           &reason;);                /* Reason code from mqexecute call      */

/*************************************************************************/
/* Check the command server is started. If not exit.                     */
/*************************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName="">\n");
   goto MOD_EXIT;
}

/*************************************************************************/
/* Check the result from mqExecute call. If successful find the channel  */
/* types for all the channels. If failed find the error.                 */
/*************************************************************************/
if ( compCode == MQCC_OK )                      /* Successful mqExecute    */
{
  /***********************************************************************/
  /* Count the number of system bags embedded in the response bag from the */
  /* mqExecute call. The attributes for each channel are in separate bags. */
  /***********************************************************************/
  mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags;,
               &compCode;, &reason;);
  CheckCallResult("Count number of bag handles", compCode, reason);

  for ( i=0; i<numberOfbags; i++)
  {
    /*********************************************************************/
    /* Get the next system bag handle out of the mqExecute response bag.   */
    /* This bag contains the channel attributes                          */
    /*********************************************************************/
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &cAttrsbag,
                 &compCode, &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);

    /*********************************************************************/
    /* Get the channel name out of the channel attributes bag              */
    /*********************************************************************/
    mqInquireString(cAttrsBag, MQCACH_CHANNEL_NAME, 0, MQ_OBJECT_NAME_LENGTH,
                    chlName, &chlNameLength, NULL, &compCode, &reason);
    CheckCallResult("Get channel name", compCode, reason);

    /*********************************************************************/
    /* Get the channel type out of the channel attributes bag              */
    /*********************************************************************/

 mqInquireInteger(cAttrsBag, MQIACH_CHANNEL_TYPE, MQIND_NONE, &chlType,
                     &compCode, &reason);
    CheckCallResult("Get type", compCode, reason);

    /*********************************************************************/
    /* Use mqTrim to prepare the channel name for printing.                */
    /* Print the result.                                                 */
    /*********************************************************************/
    mqTrim(MQ_CHANNEL_NAME_LENGTH, chlName, chlName, &compCode, &reason);
    sprintf(OutputBuffer, "%-20s%-9s", chlName, ChlType2String(chlType));
    WRITEOUTFILE(outfp,OutputBuffer,29)
  }
}

else                                             /* Failed mqExecute     */
{
  printf("Call to get channel attributes failed: Cc = %ld : Rc = %ld\n",
              compCode, reason);
  /*********************************************************************/
  /* If the command fails get the system bag handle out of the mqexecute   */
  /* response bag.This bag contains the reason from the command server     */
  /* why the command failed.                                           */
  /*********************************************************************/
  if (reason == MQRCCF_COMMAND_FAILED)
  {
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag,
                 &compCode, &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);

    /*********************************************************************/
    /* Get the completion code and reason code, returned by the command    */
    /* server, from the embedded error bag.                              */
    /*********************************************************************/
    mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
```

```
                        &compCode, &reason );
       CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
       mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                        &compCode, &reason);
       CheckCallResult("Get the reason code from the result bag",
                       compCode, reason);
       printf("Error returned by the command server: Cc = %ld : Rc = %ld\n",
                mqExecuteCC, mqExecuteRC);
     }
   }

MOD_EXIT:
   /*****************************************************************************/
   /* Delete the admin bag if successfully created.                             */
   /*****************************************************************************/
   if (adminBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&adminBag, &compCode, &reason);
      CheckCallResult("Delete the admin bag", compCode, reason);
   }

   /*****************************************************************************/
   /* Delete the response bag if successfully created.                          */
   /*****************************************************************************/
   if (responseBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&responseBag, &compCode, &reason);
      CheckCallResult("Delete the response bag", compCode, reason);
   }

   /*****************************************************************************/
   /* Disconnect from the queue manager if not already connected                */
   /*****************************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
       CheckCallResult("Disconnect from Queue Manager", compCode, reason);
   }

   /*****************************************************************************/
   /* Close the output file if open                                             */
   /*****************************************************************************/
   if(outfp != NULL)
      CLOSEOUTFILE(outfp);

   return 0;
}


/*****************************************************************************/
/*                                                                           */
/* Function: CheckCallResult                                                 */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Description of call                                    */
/*                    Completion code                                        */
/*                    Reason code                                            */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: Display the description of the call, the completion code and the   */
/*        reason code if the completion code is not successful               */
/*                                                                           */
/*****************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %ld : Reason = %ld\n", callText,
                cc, rc);
}
```

Exemple de programme C pour l'interrogation des files d'attente et des informations d'impression (amqsailq.c)

L'exemple de programme C `amqsailq.c` demande la longueur en cours des files d'attente locales à l'aide de MQAI.

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSAILQ.C                                                     */
/*                                                                              */
/* Description:  Sample C program to inquire the current depth of the local     */
/*               queues using the WebSphere MQ Administration Interface (MQAI)*/
/*                                                                              */
/* Statement:    Licensed Materials - Property of IBM                           */
/*                                                                              */
/*               84H2000, 5765-B73                                              */
/*               84H2001, 5639-B42                                              */
/*               84H2002, 5765-B74                                              */
/*               84H2003, 5765-B75                                              */
/*               84H2004, 5639-B43                                              */
/*                                                                              */
/*               (C) Copyright IBM Corp. 1999, 2024.                            */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Function:                                                                    */
/*    AMQSAILQ is a sample C program that demonstrates how to inquire           */
/*    attributes of the local queue manager using the MQAI interface. In        */
/*    particular, it inquires the current depths of all the local queues.       */
/*                                                                              */
/*     - A PCF command is built by placing items into an MQAI administration    */
/*       bag.                                                                   */
/*       These are:-                                                            */
/*            - The generic queue name "*"                                      */
/*            - The type of queue required. In this sample we want to           */
/*              inquire local queues.                                           */
/*            - The attribute to be inquired. In this sample we want the        */
/*              current depths.                                                 */
/*                                                                              */
/*     - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.       */
/*       The call generates the correct PCF structure.                          */
/*       The default options to the call are used so that the command is sent   */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                     */
/*       The reply from the command server is placed on a temporary dynamic     */
/*       queue.                                                                 */
/*       The reply from the MQCMD_INQUIRE_Q command is read from the            */
/*       temporary queue and formatted into the response bag.                   */
/*                                                                              */
/*     - The completion code from the mqExecute call is checked and if there    */
/*       is a failure from the command server, then the code returned by        */
/*       command server is retrieved from the system bag that has been          */
/*       embedded in the response bag to the mqExecute call.                    */
/*                                                                              */
/*     - If the call is successful, the depth of each local queue is placed     */
/*       in system bags embedded in the response bag of the mqExecute call.     */
/*       The name and depth of each queue is obtained from each of the bags     */
/*       and the result displayed on the screen.                                */
/*                                                                              */
/* Note: The command server must be running.                                    */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* AMQSAILQ has 1 parameter - the queue manager name (optional)                 */
/*                                                                              */
/********************************************************************************/

/********************************************************************************/
/* Includes                                                                     */
/********************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                                  */
#include <cmqcfc.h>                     /* PCF                                  */
#include <cmqbc.h>                      /* MQAI                                 */

/********************************************************************************/
/* Function prototypes                                                          */
/********************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/********************************************************************************/
/* Function: main                                                               */
/********************************************************************************/
```

```
int main(int argc, char *argv[])
{
   /*****************************************************************/
   /* MQAI variables                                                */
   /*****************************************************************/
   MQHCONN hConn;                          /* handle to WebSphere MQ connection   */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name             */
   MQLONG reason;                          /* reason code                       */
   MQLONG connReason;                      /* MQCONN reason code                */
   MQLONG compCode;                        /* completion code                   */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute           */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute         */
   MQHBAG qAttrsBag;                       /* bag containing q attributes       */
   MQHBAG errorBag;                        /* bag containing cmd server error   */
   MQLONG mqExecuteCC;                     /* mqExecute completion code         */
   MQLONG mqExecuteRC;                     /* mqExecute reason code             */
   MQLONG qNameLength;                     /* Actual length of q name           */
   MQLONG qDepth;                          /* depth of queue                    */
   MQLONG i;                               /* loop counter                      */
   MQLONG numberOfBags;                    /* number of bags in response bag    */
   MQCHAR qName[MQ_Q_NAME_LENGTH+1];       /* name of queue extracted from bag*/


   printf("Display current depths of local queues\n\n");

   /*****************************************************************/
   /* Connect to the queue manager                                  */
   /*****************************************************************/
   if (argc > 1)
      strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(qmName, &hConn, &compCode, &connReason);

   /*****************************************************************/
   /* Report the reason and stop if the connection failed.          */
   /*****************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("Queue Manager connection", compCode, connReason
);
      exit( (int)connReason);
   }

   /*****************************************************************/
   /* Create an admin bag for the mqExecute call                    */
   /*****************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &adminBag, &compCode, &reason);
   CheckCallResult("Create admin bag", compCode, reason);
   /*****************************************************************/
   /* Create a response bag for the mqExecute call                  */
   /*****************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
   CheckCallResult("Create response bag", compCode, reason);

   /*****************************************************************/
   /* Put the generic queue name into the admin bag                 */
   /*****************************************************************/
   mqAddString(adminBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, "*",
               &compCode, &reason);
   CheckCallResult("Add q name", compCode, reason);

   /*****************************************************************/
   /* Put the local queue type into the admin bag                   */
   /*****************************************************************/
   mqAddInteger(adminBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
   CheckCallResult("Add q type", compCode, reason);

   /*****************************************************************/
   /* Add an inquiry for current queue depths                       */
   /*****************************************************************/
   mqAddInquiry(adminBag, MQIA_CURRENT_Q_DEPTH, &compCode, &reason);
   CheckCallResult("Add inquiry", compCode, reason);

   /*****************************************************************/
   /* Send the command to find all the local queue names and queue depths.    */
   /* The mqExecute call creates the PCF structure required, sends it to      */
   /* the command server, and receives the reply from the command server into */
   /* the response bag. The attributes are contained in system bags that are  */
   /* embedded in the response bag, one set of attributes per bag.            */
   /*****************************************************************/
   mqExecute(hConn,                   /* WebSphere MQ connection handle           */
             MQCMD_INQUIRE_Q,         /* Command to be executed                   */
             MQHB_NONE,               /* No options bag                           */
```

```
            adminBag,               /* Handle to bag containing commands    */
            responseBag,            /* Handle to bag to receive the response*/
            MQHO_NONE,              /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
            MQHO_NONE,              /* Create a dynamic q for the response  */
            &compCode,              /* Completion code from the mqExecute   */
            &reason);               /* Reason code from mqExecute call      */


  /*********************************************************************/
  /* Check the command server is started. If not exit.                 */
  /*********************************************************************/
  if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
  {
     printf("Please start the command server: <strmqcsv QMgrName>\n");
     MQDISC(&hConn, &compCode, &reason);
     CheckCallResult("Disconnect from Queue Manager", compCode, reason);
     exit(98);
  }

  /*********************************************************************/
  /* Check the result from mqExecute call. If successful find the current    */
  /* depths of all the local queues. If failed find the error.         */
  /*********************************************************************/
  if ( compCode == MQCC_OK )                    /* Successful mqExecute    */
  {
    /*******************************************************************/
    /* Count the number of system bags embedded in the response bag from the */
    /* mqExecute call. The attributes for each queue are in a separate bag.  */
    /*******************************************************************/
    mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags, &compCode,
                 &reason);
    CheckCallResult("Count number of bag handles", compCode, reason);

    for ( i=0; i<numberOfBags; i++)
    {
      /*****************************************************************/
      /* Get the next system bag handle out of the mqExecute response bag.   */
      /* This bag contains the queue attributes                          */
      /*****************************************************************/
      mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &qAttrsBag, &compCode,
                   &reason);
      CheckCallResult("Get the result bag handle", compCode, reason);

      /*****************************************************************/
      /* Get the queue name out of the queue attributes bag                */
      /*****************************************************************/
      mqInquireString(qAttrsBag, MQCA_Q_NAME, 0, MQ_Q_NAME_LENGTH, qName,
                      &qNameLength, NULL, &compCode, &reason);
      CheckCallResult("Get queue name", compCode, reason);

      /*****************************************************************/
      /* Get the depth out of the queue attributes bag                   */
      /*****************************************************************/
      mqInquireInteger(qAttrsBag, MQIA_CURRENT_Q_DEPTH, MQIND_NONE, &qDepth,
                       &compCode, &reason);
      CheckCallResult("Get depth", compCode, reason);

      /*****************************************************************/
      /* Use mqTrim to prepare the queue name for printing.              */
      /* Print the result.                                               */
      /*****************************************************************/
      mqTrim(MQ_Q_NAME_LENGTH, qName, qName, &compCode, &reason)
      printf("%4d  %-48s\n", qDepth, qName);
    }
  }

  else                                          /* Failed mqExecute     */
  {
    printf("Call to get queue attributes failed: Completion Code = %d :
            Reason = %d\n", compCode, reason);

    /*******************************************************************/
    /* If the command fails get the system bag handle out of the mqExecute   */
    /* response bag. This bag contains the reason from the command server    */
    /* why the command failed.                                         */
    /*******************************************************************/
    if (reason == MQRCCF_COMMAND_FAILED)
    {
      mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
                   &reason);
      CheckCallResult("Get the result bag handle", compCode, reason);
```

```
      /*********************************************************************/
      /* Get the completion code and reason code, returned by the command  */
      /* server, from the embedded error bag.                              */
      /*********************************************************************/
      mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                       &compCode, &reason );
      CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
      mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                        &compCode, &reason);
      CheckCallResult("Get the reason code from the result bag",
                       compCode, reason);
      printf("Error returned by the command server: Completion Code = %d :
              Reason = %d\n", mqExecuteCC, mqExecuteRC);
    }
  }

  /*********************************************************************/
  /* Delete the admin bag if successfully created.                     */
  /*********************************************************************/
  if (adminBag != MQHB_UNUSABLE_HBAG)
  {
     mqDeleteBag(&adminBag, &compCode, &reason);
     CheckCallResult("Delete the admin bag", compCode, reason);
  }

  /*********************************************************************/
  /* Delete the response bag if successfully created.                  */
  /*********************************************************************/
  if (responseBag != MQHB_UNUSABLE_HBAG)
  {
     mqDeleteBag(&responseBag, &compCode, &reason);
     CheckCallResult("Delete the response bag", compCode, reason);
  }

  /*********************************************************************/
  /* Disconnect from the queue manager if not already connected        */
  /*********************************************************************/
  if (connReason != MQRC_ALREADY_CONNECTED)
  {
     MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("Disconnect from queue manager", compCode, reason);
  }
  return 0;
}

/*********************************************************************/
*                                                                    */
* Function: CheckCallResult                                         */
*                                                                    */
/*********************************************************************/
*                                                                    */
* Input Parameters:  Description of call                            */
*                    Completion code                                 */
*                    Reason code                                     */
*                                                                    */
* Output Parameters: None                                           */
*                                                                    */
* Logic: Display the description of the call, the completion code and the     */
*        reason code if the completion code is not successful       */
*                                                                    */
/*********************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
  if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
                callText, cc, rc);
}
```

Conseils et astuces pour la configuration de IBM WebSphere MQ

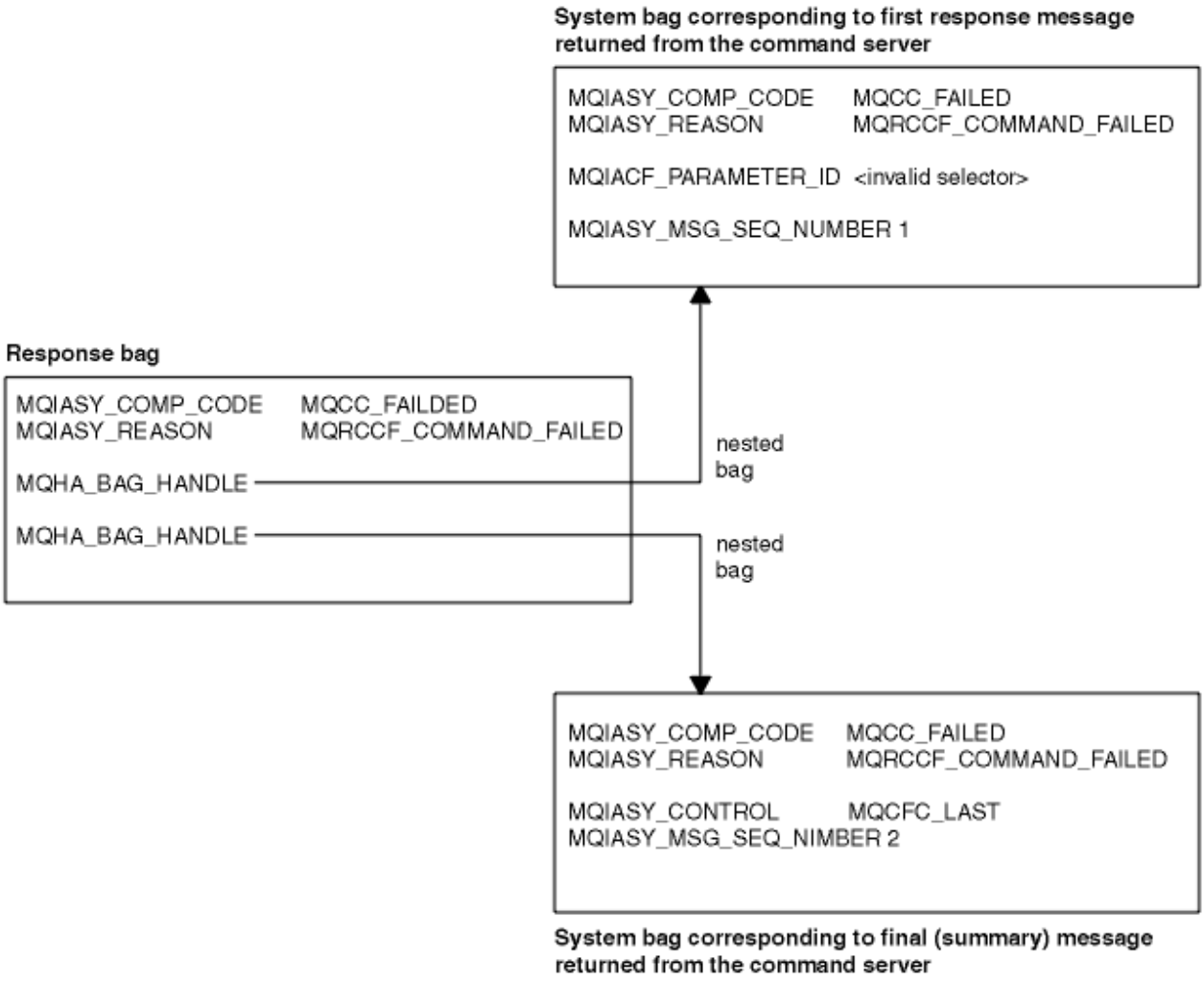
Conseils et astuces de programmation lors de l'utilisation de MQAI.

MQAI utilise des messages PCF pour envoyer des commandes d'administration au serveur de commandes au lieu de traiter directement avec le serveur de commandes lui-même. Voici quelques conseils pour la configuration de WebSphere MQ à l'aide de MQAI:

- Les chaînes de caractères dans WebSphere MQ sont complétées par des espaces de longueur fixe. A l'aide de C, les chaînes à terminaison nulle peuvent normalement être fournies en tant que paramètres d'entrée aux interfaces de programmation WebSphere MQ .
- Pour effacer la valeur d'un attribut de chaîne, définissez-la sur un seul blanc plutôt que sur une chaîne vide.
- Prenez en compte à l'avance les attributs que vous souhaitez modifier et n'examinez que ces attributs.
- Certains attributs ne peuvent pas être modifiés, par exemple un nom de file d'attente ou un type de canal. Veillez à ne modifier que les attributs qui peuvent être modifiés. Reportez-vous à la liste des paramètres obligatoires et facultatifs pour l'objet de modification PCF spécifique. Voir Définitions des formats de commande programmables.
- Si un appel MQAI échoue, certains détails de l'échec sont renvoyés au sac de réponse. Des détails supplémentaires peuvent ensuite être trouvés dans un sac imbriqué accessible par le sélecteur MQHA_BAG_HANDLE. Par exemple, si un appel mqExecute échoue avec le code anomalie MQRCCF_COMMAND_FAILED, ces informations sont renvoyées dans le sac de réponse. Ce code raison peut être dû au fait qu'un sélecteur spécifié n'était pas valide pour le type de message de commande et que ce détail d'informations se trouve dans un sac imbriqué accessible par un descripteur de sac.

  Pour plus d'informations sur MQExecute, voir «Envoi de commandes d'administration au serveur de commandes à l'aide de l'appel mqExecute», à la page 57

  Le diagramme suivant illustre ce scénario:



Rubriques MQAI avancées

Informations sur l'indexation, la conversion de données et l'utilisation du descripteur de message

- Indexation

  Les index sont utilisés lors du remplacement ou de la suppression d'éléments de données existants dans un sac afin de préserver l'ordre d'insertion. Des détails complets sur l'indexation sont disponibles dans «Indexation dans MQAI», à la page 44.

- Conversion de données

  Les chaînes contenues dans un sac de données MQAI peuvent se trouver dans divers jeux de caractères codés et peuvent être converties à l'aide de l'appel d'entier mqSet. Pour plus de détails sur la conversion des données, voir «Conversion de données dans MQAI», à la page 45.

- Utilisation du descripteur de message

  MQAI génère un descripteur de message qui est défini sur une valeur initiale lors de la création du sac de données. Vous trouverez des détails complets sur l'utilisation du descripteur de message dans «Utilisation du descripteur de message dans MQAI», à la page 46.

Indexation dans MQAI

Les index sont utilisés lors du remplacement ou de la suppression d'éléments de données existants dans un sac. Il existe trois types d'indexation, qui permettent d'extraire facilement des éléments de données.

Chaque sélecteur et chaque valeur d'un élément de données dans un sac sont associés à trois numéros d'index:

- Index relatif à d'autres éléments ayant le même sélecteur.
- Index relatif à la catégorie du sélecteur (utilisateur ou système) à laquelle appartient l'élément.
- Index relatif à tous les éléments de données du sac (utilisateur et système).

Cela permet l'indexation par des sélecteurs utilisateur, des sélecteurs système, ou les deux, comme illustré dans la Figure 1, à la page 44.

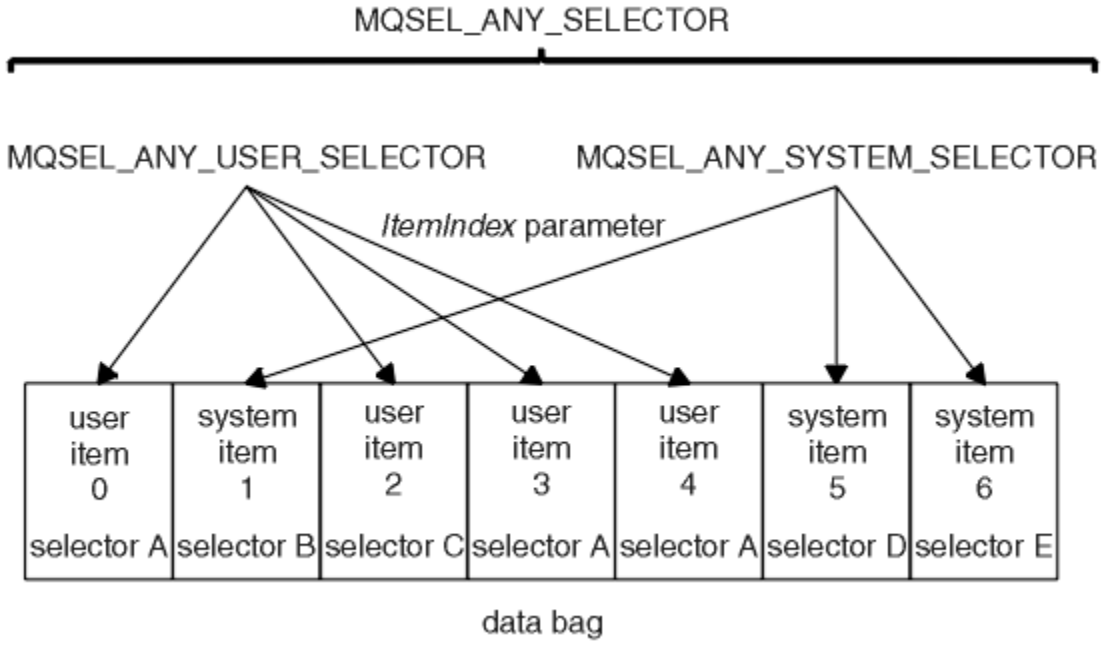


Figure 1. Indexation

Dans la figure Figure 1, à la page 44, l'élément utilisateur 3 (sélecteur A) peut être référencé par les paires d'index suivantes:

Selector	*ItemIndex*
sélecteur A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	3

L'index est basé sur zéro comme un tableau en C ; s'il y a'n'occurrences, l'index est compris entre zéro et'n-1', sans écart.

Les index sont utilisés lors du remplacement ou de la suppression d'éléments de données existants dans un sac. Lorsqu'il est utilisé de cette manière, l'ordre d'insertion est conservé, mais les index d'autres éléments de données peuvent être affectés. Pour des exemples, voir Modification des informations dans un sac et Suppression d'éléments de données.

Les trois types d'indexation permettent d'extraire facilement des éléments de données. Par exemple, s'il existe trois instances d'un sélecteur particulier dans un sac, l'appel mqCountItems peut compter le nombre d'instances de ce sélecteur, et les appels mqInquire* peuvent spécifier à la fois le sélecteur et l'index pour interroger ces valeurs uniquement. Ceci est utile pour les attributs qui peuvent avoir une liste de valeurs telles que certaines des exits sur les canaux.

Conversion de données dans MQAI

Les chaînes contenues dans un sac de données MQAI peuvent être dans une variété de jeux de caractères codés. Ces chaînes peuvent être converties à l'aide de l'appel mqSetInteger.

Comme les messages PCF, les chaînes contenues dans un sac de données MQAI peuvent être dans une variété de jeux de caractères codés. En général, toutes les chaînes d'un message PCF se trouvent dans le même jeu de caractères codés, c'est-à-dire dans le même jeu que le gestionnaire de files d'attente.

Chaque élément de chaîne d'un sac de données contient deux valeurs: la chaîne elle-même et le CCSID. La chaîne qui est ajoutée au sac est obtenue à partir du paramètre *Buffer* de la chaîne mqAddou de l'appel de chaîne mqSet. Le CCSID est obtenu à partir de l'élément système contenant un sélecteur de MQIASY_CODED_CHAR_SET_ID. Connu sous le nom de *CCSID du sac* , il peut être modifié à l'aide de l'appel mqSetInteger.

Lorsque vous interrogez la valeur d'une chaîne contenue dans un sac de données, le CCSID est un paramètre de sortie de l'appel.

Le Tableau 2, à la page 45 montre les règles appliquées lors de la conversion de sacs de données en messages et vice versa:

Tableau 2. Traitement CCSID

Appel MQAI	CCSID	Entrée à appeler	Sortie à appeler
mqBagToBuffer	CCSID du sac (1)	Ignorée	Inchangé
mqBagToBuffer	CCSID de chaîne dans le sac	Utilisé	Inchangé
mqBagToBuffer	CCSID de chaîne dans la mémoire tampon	Non applicable	Copié à partir des CCSID de chaîne dans le sac
mqBufferToBag	CCSID du sac (1)	Ignorée	Inchangé
mqBufferToBag	CCSID de chaîne dans la mémoire tampon	Utilisé	Inchangé
mqBufferToBag	CCSID de chaîne dans le sac	Non applicable	Copié à partir des CCSID de chaîne dans la mémoire tampon
SacmqPut	CCSID MQMD	Utilisé	Inchangé (2)

Tableau 2. Traitement CCSID (suite)

Appel MQAI	CCSID	Entrée à appeler	Sortie à appeler
SacmqPut	CCSID du sac (1)	Ignorée	Inchangé
SacmqPut	CCSID de chaîne dans le sac	Utilisé	Inchangé
SacmqPut	CCSID de chaîne dans le message envoyé	Non applicable	Copié à partir des CCSID de chaîne dans le sac
mqGetSac	CCSID MQMD	Utilisé pour la conversion de données de message	Défini sur le CCSID des données renvoyées (3)
mqGetSac	CCSID du sac (1)	Ignorée	Inchangé
mqGetSac	CCSID de chaîne dans le message	Utilisé	Inchangé
mqGetSac	CCSID de chaîne dans le sac	Non applicable	Copié à partir des CCSID de chaîne dans le message
mqExecute	CCSID du sac de demande	Utilisé pour le MQMD du message de demande (4)	Inchangé
mqExecute	CCSID du sac de réponse	Utilisé pour la conversion de données du message de réponse (4)	Défini sur le CCSID des données renvoyées (3)
mqExecute	CCSID de chaîne dans le sac de demande	Utilisé pour le message de demande	Inchangé
mqExecute	CCSID de chaîne dans le sac de réponse	Non applicable	Copié à partir des CCSID de chaîne dans le message de réponse

Remarques :

1. Le CCSID du sac est l'élément système avec le sélecteur MQIASY_CODED_CHAR_SET_ID.
2. MQCCSI_Q_MGR est remplacé par le CCSID réel du gestionnaire de files d'attente.
3. Si une conversion de données est demandée, le CCSID des données renvoyées est identique à la valeur de sortie. Si la conversion de données n'est pas demandée, le CCSID des données renvoyées est identique à la valeur du message. Notez qu'aucun message n'est renvoyé si la conversion de données est demandée mais échoue.
4. Si le CCSID est MQCCSI_DEFAULT, le CCSID du gestionnaire de files d'attente est utilisé.

Utilisation du descripteur de message dans MQAI

Le descripteur de message généré par MQAI est défini sur une valeur initiale lors de la création du sac de données.

Le type de commande PCF est obtenu à partir de l'élément système avec le sélecteur MQIASY_TYPE. Lorsque vous créez votre sac de données, la valeur initiale de cet élément est définie en fonction du type de sac que vous créez:

Tableau 3. Type de commande PCF	
Type de sac	**Valeur initiale de l'élément MQIASY_TYPE**
MQCBO_ADMIN_BAG	MQCFT_COMMAND (COMMANDE MQ)
SAC de commande mqcbo_commande	MQCFT_COMMAND (COMMANDE MQ)
MQCBO_ *	Utilisateur_MQCF

Lorsque MQAI génère un descripteur de message, les valeurs utilisées dans les paramètres *Format* et *MsgType* dépendent de la valeur de l'élément système avec le sélecteur MQIASY_TYPE, comme illustré dans la Tableau 3, à la page 47.

Tableau 4. Paramètres de format et MsgType de MQMD		
Type de commande PCF	**Format**	**MsgType**
MQCFT_COMMAND (COMMANDE MQ)	MQFMT_ADMIN	MQMT_REQUEST
RAPPORT MQCF	MQFMT_ADMIN	MQMT_REPORT
MQCFT_REPONSE	MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	MQFMT_ADMIN	MQMT_DATAGRAM
MQCFT_EVENT	MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_ *	MQFMT_PCF	MQMT_DATAGRAM

Tableau 4, à la page 47 montre que si vous créez un sac d'administration ou un sac de commande, le *Format* du descripteur de message est MQFMT_ADMIN et le *MsgType* est MQMT_REQUEST. Cela convient pour un message de demande PCF envoyé au serveur de commandes lorsqu'une réponse est attendue.

Les autres paramètres du descripteur de message prennent les valeurs indiquées dans le Tableau 5, à la page 47.

Tableau 5. Valeurs de descripteur de message	
Paramètre	**Valeur**
StrucId	ID_STRUCD_MQM
Version	MQMD_VERSION_1
Report	MQRO_AUCUN
MsgType	Voir Tableau 4, à la page 47
Expiry	30 secondes (remarque «1», à la page 48)
Feedback	MQFB_AUCUN
Encoding	MQENC_NATIVE
CodedCharSetId	dépend du CCSID du sac (remarque «2», à la page 48)
Format	Voir Tableau 4, à la page 47
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NON_PERSISTENT
MsgId	MQMI_AUCUN

Tableau 5. Valeurs de descripteur de message (suite)	
Paramètre	**Valeur**
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	voir la remarque «3», à la page 48
ReplyToQMgr	blanc

Remarques :

1. Cette valeur peut être remplacée dans l'appel **mqExecute** à l'aide du paramètre **OptionsBag** . Pour plus d'informations à ce sujet, voir **mqExecute**.
2. Voir «Conversion de données dans MQAI», à la page 45.
3. Nom de la file d'attente de réponses spécifiée par l'utilisateur ou de la file d'attente dynamique temporaire générée par MQAI pour les messages de type **MQMT_REQUEST**. Vide dans le cas contraire.

Sacs de données

Un sac de données est un moyen de gérer des propriétés ou des paramètres d'objets à l'aide de MQAI.

Sacs de données

- Le sac de données contient zéro ou plusieurs *éléments de données*. Ces éléments de données sont commandés dans le sac au fur et à mesure qu'ils sont placés dans le sac. Il s'agit de l' *ordre d'insertion*. Chaque élément de données contient un *sélecteur* qui identifie l'élément de données et une *valeur* de cet élément de données qui peut être un entier, un entier 64 bits, un filtre d'entier, une chaîne, un filtre de chaîne, une chaîne d'octets, un filtre de chaîne d'octets ou un descripteur d'un autre sac. Les éléments de données sont décrits en détail dans «Elément de données», à la page 51

  Il existe deux types de sélecteur: *sélecteurs d'utilisateur* et *sélecteurs de système*. Ces éléments sont décrits dans Sélecteurs MQAI. Les sélecteurs sont généralement uniques, mais il est possible d'avoir plusieurs valeurs pour le même sélecteur. Dans ce cas, un *index* identifie l'occurrence particulière du sélecteur requise. Les index sont décrits dans «Indexation dans MQAI», à la page 44.

  Une hiérarchie de ces concepts est illustrée dans la Figure 1.

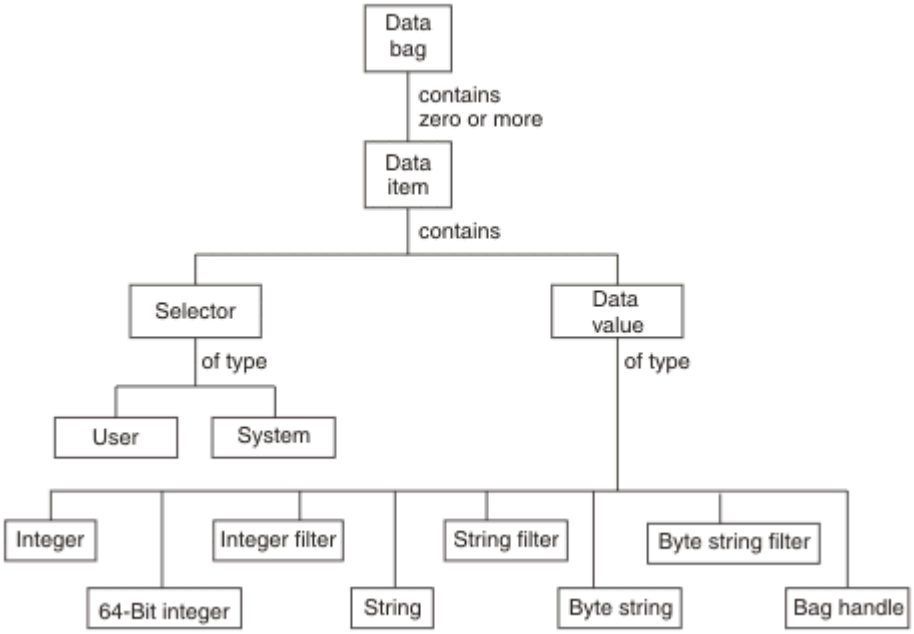


Figure 2. Hiérarchie des concepts MQAI

La hiérarchie a été expliquée dans un paragraphe précédent.

Types de sac de données

Vous pouvez choisir le type de sac de données que vous souhaitez créer en fonction de la tâche que vous souhaitez effectuer:

sac utilisateur
Sac simple utilisé pour les données utilisateur.

jeu de données d'administration
Sac créé pour les données utilisées pour administrer les objets WebSphere MQ en envoyant des messages d'administration à un serveur de commandes. Le sac d'administration implique automatiquement certaines options, comme décrit dans «Création et suppression de sacs de données», à la page 50.

sac de commandes
Un sac a également été créé pour les commandes d'administration des objets WebSphere MQ . Cependant, contrairement au sac d'administration, le sac de commande n'implique pas automatiquement certaines options bien que ces options soient disponibles. Pour plus d'informations sur les options, voir «Création et suppression de sacs de données», à la page 50.

sac de groupe
Sac utilisé pour contenir un ensemble d'éléments de données groupés. Les sacs de groupe ne peuvent pas être utilisés pour l'administration des objets WebSphere MQ .

En outre, le **sac système** est créé par MQAI lorsqu'un message de réponse est renvoyé par le serveur de commandes et placé dans le sac de sortie d'un utilisateur. Un sac système ne peut pas être modifié par l'utilisateur.

Utilisation des sacs de données Les différentes façons d'utiliser les sacs de données sont répertoriées dans cette rubrique:

Utilisation de sacs de données

Les différentes méthodes d'utilisation des sacs de données sont présentées dans la liste suivante:

- Vous pouvez créer et supprimer des sacs de données «Création et suppression de sacs de données», à la page 50.
- Vous pouvez envoyer des données entre des applications à l'aide de sacs de données «Mise en place et réception de sacs de données», à la page 51.
- Vous pouvez ajouter des éléments de données à des sacs de données «Ajout d'éléments de données à des sacs», à la page 52.
- Vous pouvez ajouter une commande d'interrogation dans un sac de données «Ajout d'une instruction d'interrogation à un sac», à la page 53.
- Vous pouvez vous renseigner dans les sacs de données «Recherche dans les sacs de données», à la page 53.
- Vous pouvez compter des éléments de données dans un sac de données «Comptage des éléments de données», à la page 56.
- Vous pouvez modifier des informations dans un sac de données «Modification d'informations dans un sac», à la page 54.
- Vous pouvez effacer un sac de données «Effacement d'un sac à l'aide de l'appel de sac mqClear», à la page 55.
- Vous pouvez tronquer un sac de données «Troncation d'un sac à l'aide de l'appel de sac mqTruncate», à la page 55.
- Vous pouvez convertir des sacs et des tampons «Conversion de sacs et de tampons», à la page 55.

Création et suppression de sacs de données

Création de sacs de données

Pour utiliser MQAI, vous devez d'abord créer un sac de données à l'aide de l'appel mqCreateBag. En entrée de cet appel, vous fournissez une ou plusieurs options pour contrôler la création du sac.

Le paramètre *Options* de l'appel MQCreateBag vous permet de choisir de créer un sac d'utilisateur, un sac de commande, un sac de groupe ou un sac d'administration.

Pour créer un sac d'utilisateur, un sac de commande ou un sac de groupe, vous pouvez choisir une ou plusieurs options supplémentaires pour:

- Utilisez le formulaire de liste lorsqu'il y a plusieurs occurrences adjacentes du même sélecteur dans un sac.
- Réorganisez les éléments de données au fur et à mesure qu'ils sont ajoutés à un message PCF pour vous assurer que les paramètres sont dans le bon ordre. Pour plus d'informations sur les éléments de données, voir «Elément de données», à la page 51.
- Vérifiez les valeurs des sélecteurs d'utilisateur pour les éléments que vous ajoutez au sac.

Les sacs d'administration impliquent automatiquement ces options.

Un sac de données est identifié par sa poignée. Le descripteur de sac est renvoyé par le sac mqCreateet doit être fourni sur tous les autres appels qui utilisent le sac de données.

Pour une description complète de l'appel de sac mqCreate, voir mqCreateBag.

Suppression de sacs de données

Tout sac de données créé par l'utilisateur doit également être supprimé à l'aide de l'appel de sac mqDelete. Par exemple, si un sac est créé dans le code utilisateur, il doit également être supprimé dans le code utilisateur.

Les sacs système sont créés et supprimés automatiquement par MQAI. Pour plus d'informations à ce sujet, voir «Envoi de commandes d'administration au serveur de commandes à l'aide de l'appel mqExecute», à la page 57. Le code utilisateur ne peut pas supprimer un sac système.

Pour une description complète de l'appel de sac mqDelete, voir mqDeleteBag.

Mise en place et réception de sacs de données

Les données peuvent également être envoyées entre les applications en plaçant et en obtenant des sacs de données à l'aide des appels de sac mqPutet mqGet. Cela permet à MQAI de gérer la mémoire tampon plutôt que l'application. L'appel de sac mqPutconvertit le contenu du sac spécifié en message PCF et envoie le message à la file d'attente spécifiée ; l'appel de sac mqGetsupprime le message de la file d'attente spécifiée et le reconvertit en sac de données. Par conséquent, l'appel mqPutest l'équivalent de l'appel mqBagToBuffer suivi de MQPUT, et l'appel mqGetBag est l'équivalent de l'appel MQGET suivi de mqBufferToBag.

Pour plus d'informations sur l'envoi et la réception de messages PCF dans une file d'attente spécifique, voir «Envoi et réception de messages PCF dans une file d'attente spécifiée», à la page 12

Remarque : Si vous choisissez d'utiliser l'appel mqGetBag, les détails PCF dans le message doivent être corrects ; si ce n'est pas le cas, une erreur appropriée se produit et le message PCF n'est pas renvoyé.

Elément de données

Les éléments de données sont utilisés pour remplir les sacs de données lors de leur création. Ces éléments de données peuvent être des éléments utilisateur ou système.

Ces éléments utilisateur contiennent des données utilisateur telles que les attributs des objets en cours de gestion. Les éléments système doivent être utilisés pour mieux contrôler les messages générés: par exemple, la génération d'en-têtes de message. Pour plus d'informations sur les éléments système, voir «Eléments système», à la page 51.

Types d'éléments de données

Une fois que vous avez créé un sac de données, vous pouvez le remplir avec des éléments de type entier ou chaîne de caractères. Vous pouvez vous renseigner sur les trois types d'élément.

L'élément de données peut être un entier ou une chaîne de caractères. Voici les types d'élément de données disponibles dans MQAI:

- Entier
- Entier 64 bits
- Filtre de type entier
- Chaîne de caractères
- Filtre de chaîne
- Chaîne d'octets
- Filtre de chaîne d'octets
- Poignée de sac

Utilisation d'éléments de données

Les méthodes d'utilisation des éléments de données sont les suivantes:

- «Comptage des éléments de données», à la page 56.
- «Suppression d'éléments de données», à la page 56.
- «Ajout d'éléments de données à des sacs», à la page 52.
- «Filtrage et interrogation des éléments de données», à la page 53.

Eléments système

Les éléments système peuvent être utilisés pour:

- Génération des en-têtes PCF. Les éléments système peuvent contrôler l'identificateur de commande PCF, les options de contrôle, le numéro de séquence de message et le type de commande.

- Conversion de données. Les éléments système gèrent l'identificateur de jeu de caractères pour les éléments de chaîne de caractères du sac.

Comme tous les éléments de données, les éléments système sont constitués d'un sélecteur et d'une valeur. Pour plus d'informations sur ces sélecteurs et sur leur fonction, voir Sélecteurs MQAI.

Les éléments système sont uniques. Un ou plusieurs éléments système peuvent être identifiés par un sélecteur de système. Il n'y a qu'une seule occurrence de chaque sélecteur de système.

La plupart des éléments système peuvent être modifiés (voir «Modification d'informations dans un sac», à la page 54), mais les options de création de sac ne peuvent pas être modifiées par l'utilisateur. Vous ne pouvez pas supprimer d'éléments système. (Voir «Suppression d'éléments de données», à la page 56.)

Ajout d'éléments de données à des sacs

Lorsqu'un sac de données est créé, vous pouvez le remplir avec des éléments de données. Ces éléments de données peuvent être des éléments utilisateur ou système. Pour plus d'informations sur les éléments de données, voir «Elément de données», à la page 51.

MQAI vous permet d'ajouter des éléments de type entier, des éléments de type entier 64 bits, des éléments de filtre de type entier, des éléments de type chaîne de caractères, des éléments de filtre de type chaîne de caractères, des éléments de filtre de type chaîne d'octets et des éléments de filtre de type chaîne d'octets à des sacs, comme illustré dans la Figure 3, à la page 52. Les éléments sont identifiés par un sélecteur. Généralement, un sélecteur identifie un seul élément, mais ce n'est pas toujours le cas. Si un élément de données avec le sélecteur spécifié est déjà présent dans le sac, une instance supplémentaire de ce sélecteur est ajoutée à la fin du sac.

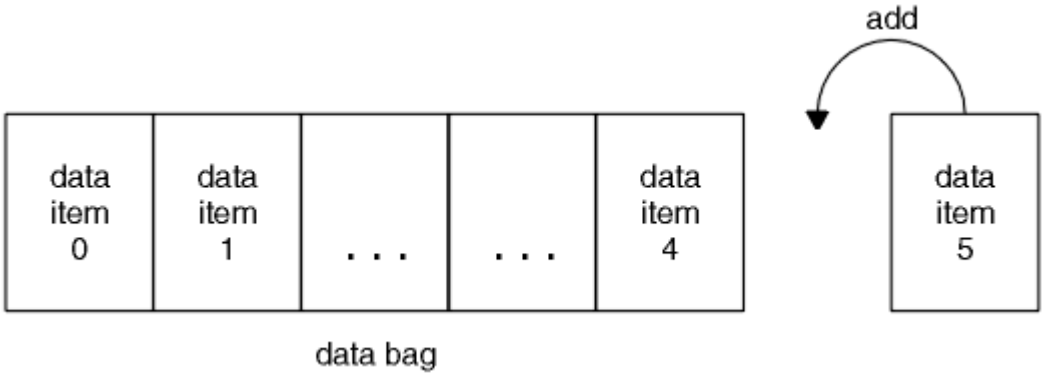


Figure 3. Ajout d'éléments de données

Ajoutez des éléments de données à un sac à l'aide des appels mqAdd*:

- Pour ajouter des éléments de type entier, utilisez l'appel mqAddInteger comme décrit dans mqAddInteger
- Pour ajouter des entiers 64 bits, utilisez l'appel mqAddInteger64 comme décrit dans mqAddInteger64
- Pour ajouter des éléments de filtre de type entier, utilisez l'appel mqAddIntegerFilter comme décrit dans mqAddIntegerFilter
- Pour ajouter des éléments de chaîne de caractères, utilisez l'appel de chaîne mqAddcomme décrit dans mqAddString
- Pour ajouter des éléments de filtre de chaîne, utilisez l'appel mqAddStringFilter comme décrit dans mqAddStringFilter
- Pour ajouter des éléments de chaîne d'octets, utilisez l'appel mqAddByteString comme décrit dans mqAddByteString
- Pour ajouter des éléments de filtre de chaîne d'octets, utilisez l'appel de filtre mqAddByteStringcomme décrit dans mqAddByteStringFilter

Pour plus d'informations sur l'ajout d'éléments de données à un sac, voir «Eléments système», à la page 51.

Ajout d'une instruction d'interrogation à un sac

L'appel d'interrogation mqAddpermet d'ajouter une commande d'interrogation à un sac. L'appel est spécifique à des fins d'administration, de sorte qu'il peut être utilisé avec des sacs d'administration uniquement. Il vous permet de spécifier les sélecteurs d'attributs sur lesquels vous souhaitez vous renseigner à partir de WebSphere MQ.

Pour une description complète de l'appel mqAddInquiry, voir mqAddInquiry.

Filtrage et interrogation des éléments de données

Lorsque vous utilisez MQAI pour vous renseigner sur les attributs des objets WebSphere MQ , vous pouvez contrôler les données renvoyées à votre programme de deux manières.

- Vous pouvez ***filtrer*** les données renvoyées à l'aide des appels de chaîne mqAddInteger et mqAdd. Cette approche vous permet de spécifier une paire *Selector* et *ItemValue* , par exemple:

  ```
  mqAddInteger(inputbag, MQIA_Q_TYPE, MQQT_LOCAL)
  ```

  Cet exemple indique que le type de file d'attente (*Selector*) doit être local (*ItemValue*) et que cette spécification doit correspondre aux attributs de l'objet (dans ce cas, une file d'attente) que vous interrogez.

  Les autres attributs qui peuvent être filtrés correspondent aux commandes PCF Inquire * qui se trouvent dans «Présentation des commandes PCF (Programmable Command Formats)», à la page 9. Par exemple, pour en savoir plus sur les attributs d'un canal, reportez-vous à la commande Inquire Channel dans la documentation du produit. Les "paramètres requis" et les "paramètres facultatifs" de la commande Inquire Channel identifient les sélecteurs que vous pouvez utiliser pour le filtrage.
- Vous pouvez ***interroger*** les attributs particuliers d'un objet à l'aide de l'appel d'interrogation mqAdd. Indique le sélecteur qui vous intéresse. Si vous ne spécifiez pas le sélecteur, tous les attributs de l'objet sont renvoyés.

Voici un exemple de filtrage et d'interrogation des attributs d'une file d'attente:

```
/* Request information about all queues */
mqAddString(adminbag, MQCA_Q_NAME, "*")

/* Filter attributes so that local queues only are returned */
mqAddInteger(adminbag, MQIA_Q_TYPE, MQQT_LOCAL)

/* Query the names and current depths of the local queues */
mqAddInquiry(adminbag, MQCA_Q_NAME)
mqAddInquiry(adminbag, MQIA_CURRENT_Q_DEPTH)

/* Send inquiry to the command server and wait for reply */
mqExecute(MQCMD_INQUIRE_Q, ...)
```

Pour obtenir d'autres exemples de filtrage et d'interrogation des éléments de données, voir «Exemples d'utilisation de MQAI», à la page 22.

Recherche dans les sacs de données

Vous pouvez vous renseigner sur:

- Valeur d'un élément entier à l'aide de l'appel mqInquireInteger. Voir mqInquireInteger.
- Valeur d'un élément entier 64 bits à l'aide de l'appel mqInquireInteger64 . Voir mqInquireInteger64.
- Valeur d'un élément de filtre d'entier à l'aide de l'appel mqInquireIntegerFilter . Voir mqInquireIntegerFilter.
- Valeur d'un élément de chaîne de caractères à l'aide de l'appel de chaîne mqInquire. Voir mqInquireString.
- Valeur d'un élément de filtre de chaîne à l'aide de l'appelStringFilter mqInquire. Voir mqInquireStringFilter.

- Valeur d'un élément de chaîne d'octets à l'aide de l'appel mqInquireByteString . Voir mqInquireByteString.
- Valeur d'un élément de filtre de chaîne d'octet à l'aide de l'appel de filtre mqInquireByteString. Voir mqInquireByteStringFilter.
- Valeur d'un descripteur de sac à l'aide de l'appel de sac mqInquire. Voir mqInquireBag.

Vous pouvez également vous renseigner sur le type (entier, entier 64 bits, filtre entier, chaîne de caractères, filtre de chaîne, chaîne d'octets, filtre de chaîne d'octets ou descripteur de sac) d'un élément spécifique à l'aide de l'appel mqInquireItemInfo . Voir mqInquireItemInfo.

Modification d'informations dans un sac

L'interface MQAI vous permet de modifier des informations dans un sac à l'aide des appels mqSet*. Vous pouvez :

1. Modifier des éléments de données dans un sac. L'index permet de remplacer une instance individuelle d'un paramètre en identifiant l'occurrence de l'élément à modifier (voir Figure 4, à la page 54).

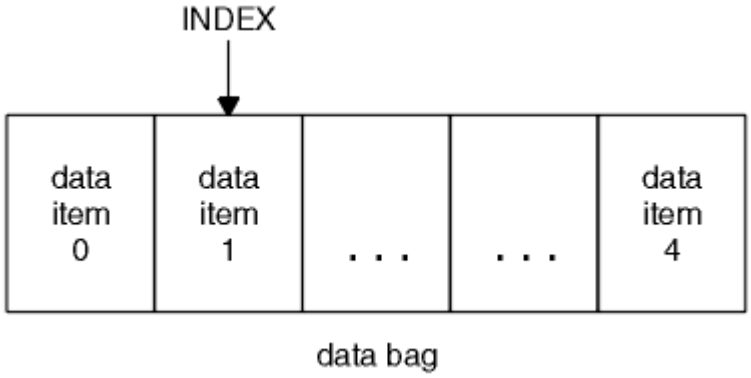


Figure 4. Modification d'un élément de données unique

2. Supprimez toutes les occurrences existantes du sélecteur spécifié et ajoutez une nouvelle occurrence à la fin du sac. (Voir Figure 5, à la page 54.) Une valeur d'index spéciale permet de remplacer ***toutes les instances*** d'un paramètre.

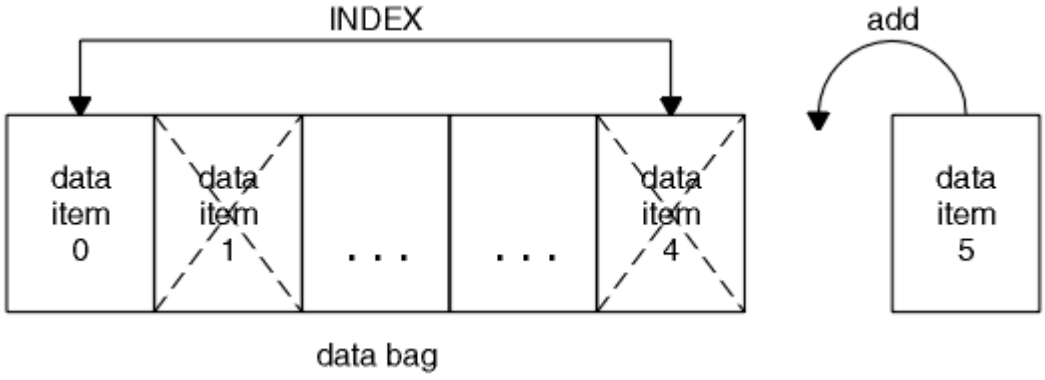


Figure 5. Modification de tous les éléments de données

Remarque : L'index conserve l'ordre d'insertion dans le sac mais peut affecter les index d'autres éléments de données.

L'appel mqSetInteger vous permet de modifier des éléments de type entier dans un sac. L'appel mqSetInteger64 vous permet de modifier des éléments d'entier 64 bits. L'appel mqSetIntegerFilter vous permet de modifier des éléments de filtre de type entier. L'appel de chaîne mqSetvous permet de modifier des éléments de chaîne de caractères. L'appel mqSetStringFilter vous permet de modifier des éléments de filtre de chaîne. L'appel mqSetByteString vous permet de modifier des éléments de chaîne d'octets. L'appel de filtre mqSetByteStringvous permet de modifier des éléments de filtre de chaîne d'octets. Vous pouvez également utiliser ces appels pour supprimer toutes les occurrences existantes du sélecteur

spécifié et ajouter une nouvelle occurrence à la fin du sac. L'élément de données peut être un élément utilisateur ou un élément système.

Pour une description complète de ces appels, voir:

- mqSetentier
- mqSetInteger64
- mqSetIntegerFilter
- mqSetChaîne
- mqSetStringFilter
- mqSetByteString
- mqSetByteStringFiltre

Effacement d'un sac à l'aide de l'appel de sac mqClear

L'appel de sac mqClearsupprime tous les éléments utilisateur d'un sac utilisateur et réinitialise les éléments système à leurs valeurs initiales. Les sacs système contenus dans le sac sont également supprimés.

Pour une description complète de l'appel de sac mqClear, voir mqClearBag.

Troncation d'un sac à l'aide de l'appel de sac mqTruncate

L'appel de sac mqTruncateréduit le nombre d'éléments utilisateur dans un sac utilisateur en supprimant les éléments de la fin du sac, en commençant par l'élément ajouté le plus récemment. Par exemple, il peut être utilisé lors de l'utilisation des mêmes informations d'en-tête pour générer plusieurs messages.

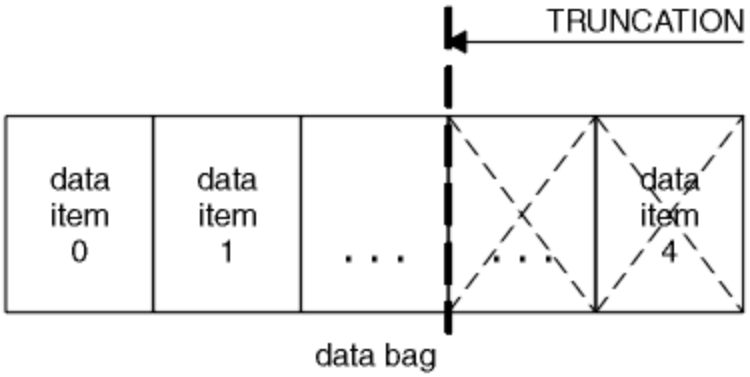


Figure 6. Troncation d'un sac

Pour une description complète de l'appel de sac mqTruncate, voir mqTruncateBag.

Conversion de sacs et de tampons

Pour envoyer des données entre applications, on place d'abord les données de message dans un sac. Ensuite, les données du sac sont converties en message PCF à l'aide de l'appelToBuffer mqBag. Le message PCF est envoyé à la file d'attente requise à l'aide de l'appel MQPUT. Ceci est illustré dans la figure Figure 7, à la page 55. Pour une description complète de l'appel mqBagToBuffer , voir mqBagToBuffer.

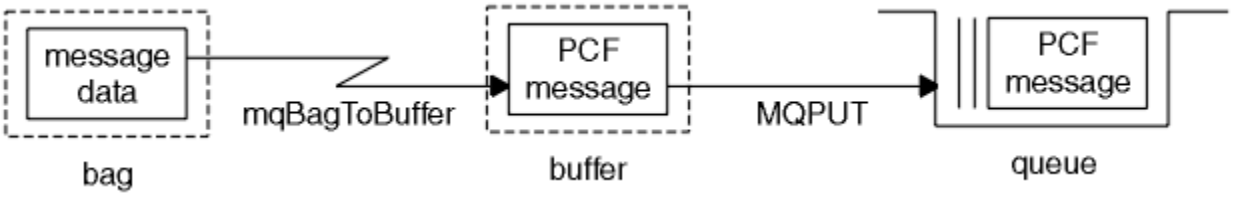


Figure 7. Conversion de sacs en messages PCF

Pour recevoir des données, le message est reçu dans une mémoire tampon à l'aide de l'appel MQGET. Les données de la mémoire tampon sont ensuite converties en un sac à l'aide de l'appelToBag mqBuffer, à condition que la mémoire tampon contienne un message PCF valide. Cela est illustré dans la figure Figure 8, à la page 56. Pour une description complète de l'appel mqBufferToBag , voir mqBufferToBag.

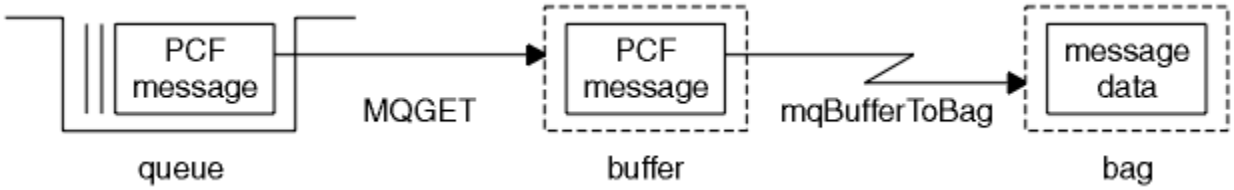


Figure 8. Conversion de messages PCF en forme de sac

Comptage des éléments de données

L'appel d'éléments mqCountcompte le nombre d'éléments utilisateur, d'éléments système, ou les deux, qui sont stockés dans un sac de données et renvoie ce nombre. Par exemple, `mqCountItems(`*Bag*`, 7, ...)` renvoie le nombre d'éléments dans le sac avec un sélecteur de 7. Il peut compter des éléments par sélecteur individuel, par sélecteur d'utilisateur, par sélecteur système ou par tous les sélecteurs.

Remarque : Cet appel compte le nombre d'éléments de données, et non le nombre de sélecteurs uniques dans le sac. Un sélecteur peut se produire plusieurs fois, de sorte qu'il peut y avoir moins de sélecteurs uniques dans le sac que d'éléments de données.

Pour une description complète de l'appel mqCountItems, voir mqCountItems.

Suppression d'éléments de données

Vous pouvez supprimer des éléments de sacs de plusieurs manières. Vous pouvez :

- Retirez un ou plusieurs éléments utilisateur d'un sac. Pour des informations détaillées, voir «Suppression d'éléments de données d'un sac à l'aide de l'appel d'élément mqDelete», à la page 56.
- Supprimez ***tous*** les éléments utilisateur d'un sac, c'est-à-dire *désélectionnez* un sac. Pour plus d'informations, voir «Effacement d'un sac à l'aide de l'appel de sac mqClear», à la page 55.
- Supprimer des éléments utilisateur de la fin d'un sac, c'est-à-dire *tronquer* un sac. Pour des informations détaillées, voir «Troncation d'un sac à l'aide de l'appel de sac mqTruncate», à la page 55.

Suppression d'éléments de données d'un sac à l'aide de l'appel d'élément mqDelete

L'appel d'élément mqDeletesupprime un ou plusieurs éléments utilisateur d'un sac. L'index est utilisé pour supprimer:

1. Une seule occurrence du sélecteur spécifié. (Voir Figure 9, à la page 56.)

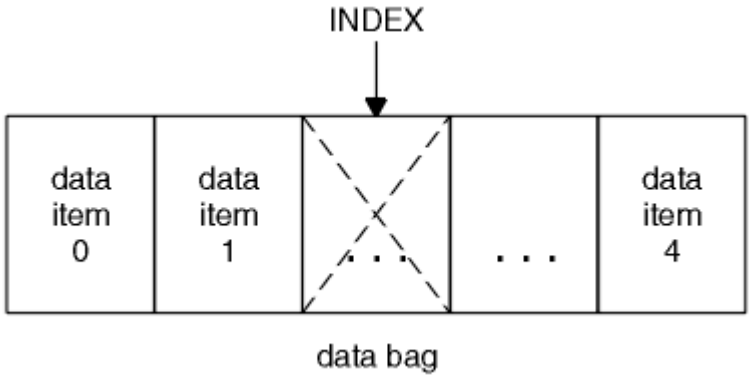


Figure 9. Suppression d'un élément de données unique

ou

2. Toutes les occurrences du sélecteur spécifié. (Voir Figure 10, à la page 57.)

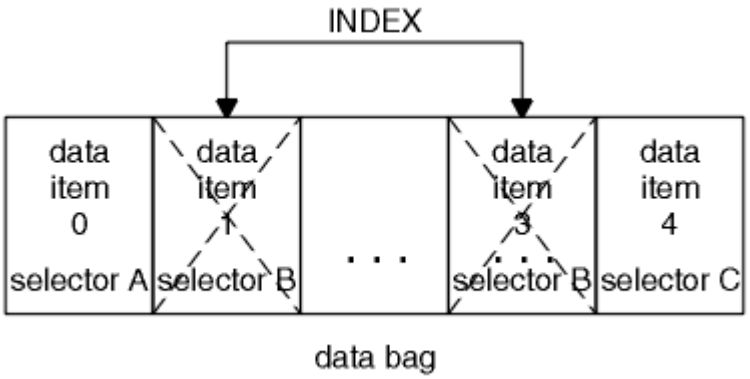


Figure 10. Suppression de tous les éléments de données

Remarque : L'index conserve l'ordre d'insertion dans le sac mais peut affecter les index d'autres éléments de données. Par exemple, l'appel d'élément mqDeletene conserve pas les valeurs d'index des éléments de données qui suivent l'élément supprimé car les index sont réorganisés pour combler l'écart qui reste de l'élément supprimé.

Pour une description complète de l'appel d'élément mqDelete, voir mqDeleteItem.

Envoi de commandes d'administration au serveur de commandes à l'aide de l'appel mqExecute

Lorsqu'un sac de données a été créé et rempli, un message de commande d'administration peut être envoyé au serveur de commandes d'un gestionnaire de files d'attente à l'aide de l'appel mqExecute . Cela permet de gérer l'échange avec le serveur de commandes et de renvoyer les réponses dans un sac.

Une fois que vous avez créé et rempli votre sac de données, vous pouvez envoyer un message de commande d'administration au serveur de commandes d'un gestionnaire de files d'attente. La méthode la plus simple consiste à utiliser l'appel mqExecute . L'appel mqExecute envoie un message de commande d'administration en tant que message non persistant et attend les réponses. Les réponses sont renvoyées dans un sac de réponse. Ils peuvent contenir des informations sur les attributs relatifs à plusieurs objets WebSphere MQ ou à une série de messages de réponse d'erreur PCF, par exemple. Par conséquent, le sac de réponse peut contenir un code retour uniquement ou *des sacs imbriqués*.

Les messages de réponse sont placés dans des sacs système créés par le système. Par exemple, pour les demandes concernant les noms d'objets, un sac système est créé pour contenir ces noms d'objet et le sac est inséré dans le sac utilisateur. Les poignées de ces sacs sont ensuite insérées dans le sac de réponse et le sac imbriqué est accessible par le sélecteur MQHA_BAG_HANDLE. Le sac système reste en stockage, s'il n'est pas supprimé, jusqu'à ce que le sac de réponse soit supprimé.

Le concept d' *imbrication* est illustré dans la Figure 11, à la page 58.

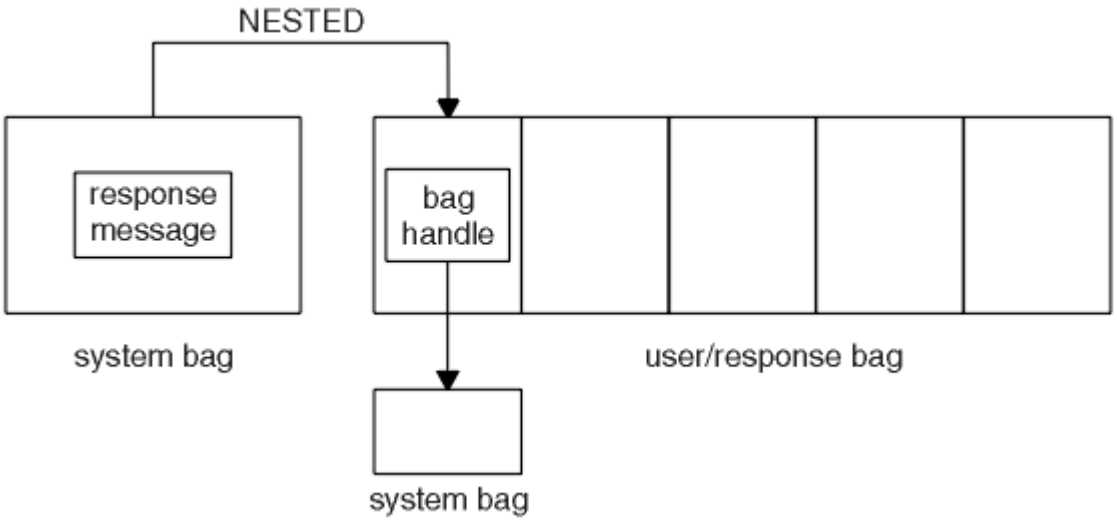


Figure 11. Imbrication

En tant qu'entrée de l'appel mqExecute , vous devez fournir:

- Un descripteur de connexion MQI.
- Commande à exécuter. Il doit s'agir de l'une des valeurs MQCMD_ *.

  Remarque : Si cette valeur n'est pas reconnue par MQAI, elle est toujours acceptée. Toutefois, si l'appel d'interrogation mqAdda été utilisé pour insérer des valeurs dans le sac, ce paramètre doit être une commande INQUIRE reconnue par MQAI. Autrement dit, le paramètre doit être de la forme MQCMD_INQUIRE_ *.
- Eventuellement, une poignée du sac contenant des options qui contrôlent le traitement de l'appel. C'est également là que vous pouvez spécifier la durée maximale en millisecondes pendant laquelle MQAI doit attendre chaque message de réponse.
- Descripteur du sac d'administration qui contient les détails de la commande d'administration à émettre.
- Descripteur du sac de réponse qui reçoit les messages de réponse.

Les éléments suivants sont facultatifs:

- Descripteur d'objet de la file d'attente dans laquelle la commande d'administration doit être placée.

  Si aucun descripteur d'objet n'est spécifié, la commande d'administration est placée sur SYSTEM.ADMIN.COMMAND.QUEUE appartenant au gestionnaire de files d'attente actuellement connecté. Il s'agit de l'option par défaut.
- Descripteur d'objet de la file d'attente dans laquelle les messages de réponse doivent être placés.

  Vous pouvez choisir de placer les messages de réponse dans une file d'attente dynamique créée automatiquement par MQAI. La file d'attente créée existe uniquement pour la durée de l'appel et est supprimée par l'interface MQAI à la sortie de l'appel mqExecute .

Pour obtenir des exemples d'utilisation de l'appel mqExecute , voir Exemple de code

Administration à l'aide de IBM WebSphere MQ Explorer

IBM WebSphere MQ Explorer vous permet d'effectuer l'administration locale ou à distance de votre réseau à partir d'un ordinateur exécutant Windowsou Linux (plateformesx86 et x86-64) uniquement.

IBM WebSphere MQ for Windowset IBM WebSphere MQ for Linux (plateformesx86 et x86-64) fournissent une interface d'administration appelée IBM WebSphere MQ Explorer pour effectuer des tâches d'administration au lieu d'utiliser des commandes de contrôle ou MQSC. La comparaison des ensembles de commandes vous montre ce que vous pouvez faire à l'aide de l' IBM WebSphere MQ Explorer.

IBM WebSphere MQ Explorer vous permet d'effectuer une administration locale ou à distance de votre réseau à partir d'un ordinateur exécutant Windowsou Linux (plateformesx86-64), en pointant le IBM WebSphere MQ Explorer vers les gestionnaires de files d'attente et les clusters qui vous intéressent. Les plateformes et les niveaux de IBM WebSphere MQ qui peuvent être administrés à l'aide de l' IBM WebSphere MQ Explorer sont décrits dans «Gestionnaires de files d'attente éloignées», à la page 60.

Pour configurer les gestionnaires de files d'attente IBM WebSphere MQ distantes afin que IBM WebSphere MQ Explorer puisse les administrer, voir «Logiciels prérequis et définitions», à la page 61.

Il vous permet d'effectuer des tâches, généralement associées à la configuration et à l'optimisation de l'environnement de travail pour IBM WebSphere MQ, localement ou à distance dans un domaine système Windows ou Linux (plateformesx86 et x86-64).

Sous Linux, le démarrage de IBM WebSphere MQ Explorer peut échouer si vous disposez de plusieurs installations Eclipse . Dans ce cas, démarrez IBM WebSphere MQ Explorer à l'aide d'un ID utilisateur différent de celui que vous utilisez pour l'autre installation Eclipse .

Sous Linux, pour démarrer correctement IBM WebSphere MQ Explorer , vous devez pouvoir écrire un fichier dans votre répertoire de base et ce dernier doit exister.

Ce que vous pouvez faire avec le IBM WebSphere MQ Explorer

Il s'agit de la liste des tâches que vous pouvez effectuer à l'aide de l' IBM WebSphere MQ Explorer.

Avec l'explorateur IBM WebSphere MQ , vous pouvez:

- Créez et supprimez un gestionnaire de files d'attente (sur votre machine locale uniquement).
- Démarrez et arrêtez un gestionnaire de files d'attente (sur votre machine locale uniquement).
- Définissez, affichez et modifiez les définitions des objets WebSphere MQ tels que les files d'attente et les canaux.
- Parcourez les messages d'une file d'attente.
- Démarrer et arrêter un canal.
- Afficher des informations de statut sur un canal, un programme d'écoute, une file d'attente ou des objets de service.
- Afficher les gestionnaires de files d'attente dans un cluster.
- Vérifiez les applications, les utilisateurs ou les canaux pour lesquels une file d'attente particulière est ouverte.
- Créez un cluster de gestionnaires de files d'attente à l'aide de l'assistant *Créer un cluster* .
- Ajoutez un gestionnaire de files d'attente à un cluster à l'aide de l'assistant *Ajout d'un gestionnaire de files d'attente à un cluster* .
- Gérez l'objet d'informations d'authentification, utilisé avec la sécurité du canal SSL (Secure Sockets Layer).
- Créer et supprimer des initiateurs de canal, des moniteurs de déclenchement et des programmes d'écoute.
- Démarrez ou arrêtez les serveurs de commandes, les initiateurs de canal, les moniteurs de déclenchement et les programmes d'écoute.
- Définissez des services spécifiques pour qu'ils démarrent automatiquement lorsqu'un gestionnaire de files d'attente est démarré.
- Modifiez les propriétés des gestionnaires de files d'attente.
- Modifiez le gestionnaire de files d'attente par défaut local.
- Appelez l'interface graphique d'ikeyman pour gérer les certificats SSL (Secure Sockets Layer), associer des certificats à des gestionnaires de files d'attente et configurer des magasins de certificats (sur votre machine locale uniquement).
- Créez des objets JMS à partir d'objets WebSphere MQ et des objets WebSphere MQ à partir d'objets JMS.

- Créez une fabrique de connexions JMS pour l'un des types actuellement pris en charge.
- Modifiez les paramètres d'un service, comme le numéro de port TCP d'un programme d'écoute ou le nom d'une file d'attente d'initiateur de canal.
- Démarrez ou arrêtez la trace de service.

Vous effectuez des tâches d'administration à l'aide d'une série de *Vues de contenu* et de *boîtes de dialogue de propriété*.

Vue Contenu
Une vue de contenu est un panneau qui peut afficher les éléments suivants:

- Attributs et options d'administration liés à WebSphere MQ lui-même.
- Attributs et options d'administration relatifs à un ou plusieurs objets associés.
- Attributs et options d'administration d'un cluster.

Boîtes de dialogue de propriétés
Une boîte de dialogue de propriété est un panneau qui affiche les attributs relatifs à un objet dans une série de zones, dont certaines peuvent être éditées.

Vous naviguez dans WebSphere MQ Explorer à l'aide de la vue *Navigator*. Le Navigator vous permet de sélectionner la vue de contenu dont vous avez besoin.

Gestionnaires de files d'attente éloignées

Il existe deux exceptions aux gestionnaires de files d'attente pris en charge auxquels vous pouvez vous connecter.

A partir d'un système Windows ou Linux (plateformesx86 et x86-64), WebSphere MQ Explorer peut se connecter à tous les gestionnaires de files d'attente pris en charge avec les exceptions suivantes:

- WebSphere MQ for z/OS gestionnaires de files d'attente antérieurs à la version 6.0.
- Gestionnaires de files d'attente MQSeries V2 actuellement pris en charge.

IBM WebSphere MQ Explorer gère les différences dans les capacités entre les différents niveaux de commande et les différentes plateformes. Toutefois, s'il rencontre un attribut qu'il ne reconnaît pas, l'attribut ne sera pas visible.

Si vous prévoyez d'administrer à distance un gestionnaire de files d'attente V6.0 ou ultérieure sous Windows à l'aide de l'explorateur IBM WebSphere MQ sur un ordinateur WebSphere MQ V5.3 , vous devez installer le groupe de correctifs 9 (CSD9) ou ultérieur sur votre ordinateur WebSphere MQ for Windows V5.3 .

Si vous prévoyez d'administrer à distance un gestionnaire de files d'attente V5.3 sur iSeries à l'aide de WebSphere MQ Explorer sur un ordinateur WebSphere MQ V6.0 ou ultérieure, vous devez installer le groupe de correctifs 11 (CSD11) ou une version ultérieure sur votre ordinateur WebSphere MQ for iSeries V5.3 . Ce groupe de correctifs corrige les problèmes de connexion entre WebSphere MQ Explorer et le gestionnaire de files d'attente iSeries .

Choix de l'utilisation du IBM WebSphere MQ Explorer

Lorsque vous décidez d'utiliser le IBM WebSphere MQ Explorer lors de votre installation, prenez en compte les informations répertoriées dans cette rubrique.

Vous devez prendre en compte les points suivants:

Noms d'objet
Si vous utilisez des noms en minuscules pour les gestionnaires de files d'attente et d'autres objets avec IBM WebSphere MQ Explorer, lorsque vous utilisez les objets à l'aide de commandes MQSC, vous devez les placer entre guillemets simples, sinon WebSphere MQ ne les reconnaît pas.

Grands gestionnaires de files d'attente
IBM WebSphere MQ Explorer fonctionne mieux avec les petits gestionnaires de files d'attente. Si vous disposez d'un grand nombre d'objets sur un seul gestionnaire de files d'attente, vous risquez de

rencontrer des retards pendant que l'explorateur WebSphere MQ extrait les informations requises à présenter dans une vue.

Groupes
WebSphere Les clusters MQ peuvent potentiellement contenir des centaines ou des milliers de gestionnaires de files d'attente. WebSphere MQ Explorer présente les gestionnaires de files d'attente dans un cluster à l'aide d'une structure arborescente. La taille physique d'un cluster n'affecte pas considérablement la vitesse de l'explorateur IBM WebSphere MQ car l'explorateur IBM WebSphere MQ ne se connecte pas aux gestionnaires de files d'attente du cluster tant que vous ne les avez pas sélectionnés.

Configuration de IBM WebSphere MQ Explorer

Cette section décrit les étapes à suivre pour configurer IBM WebSphere MQ Explorer.

Logiciels prérequis et définitions

Vérifiez que vous respectez les exigences suivantes avant de tenter d'utiliser le IBM WebSphere MQ Explorer.

IBM WebSphere MQ Explorer peut se connecter à des gestionnaires de files d'attente éloignées à l'aide du protocole de communication TCP/IP uniquement.

Vérifiez que:

1. Un serveur de commandes est en cours d'exécution sur chaque gestionnaire de files d'attente administré à distance.
2. Un objet programme d'écoute TCP/IP approprié doit être en cours d'exécution sur chaque gestionnaire de files d'attente éloignées. Cet objet peut être le programme d'écoute IBM WebSphere MQ ou, sur les systèmes UNIX and Linux , le démon inetd.
3. Un canal de connexion serveur, par défaut nommé SYSTEM.ADMIN.SVRCONN, existe sur tous les gestionnaires de files d'attente éloignées.

   Vous pouvez créer le canal à l'aide de la commande MQSC suivante:

   ```
   DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   ```

   Cette commande crée une définition de canal de base. Si vous souhaitez une définition plus sophistiquée (pour configurer la sécurité, par exemple), vous avez besoin de paramètres supplémentaires. Pour plus d'informations, voir DEFINE CHANNEL .
4. La file d'attente système, SYSTEM.MQEXPLORER.REPLY.MODEL, doit exister.

Sécurité

Si vous utilisez WebSphere MQ dans un environnement où il est important que vous contrôliez l'accès des utilisateurs à des objets particuliers, vous devrez peut-être prendre en compte les aspects de sécurité de l'utilisation de IBM WebSphere MQ Explorer.

Autorisation d'utiliser le IBM WebSphere MQ Explorer

Tout utilisateur peut utiliser le IBM WebSphere MQ Explorer, mais certains droits sont requis pour la connexion, l'accès et la gestion des gestionnaires de files d'attente.

Pour effectuer des tâches d'administration locales à l'aide de WebSphere MQ Explorer, un utilisateur doit disposer des droits nécessaires pour effectuer les tâches d'administration. Si l'utilisateur est membre du groupe mqm , il est autorisé à effectuer toutes les tâches d'administration locales.

Pour se connecter à un gestionnaire de files d'attente éloignées et effectuer des tâches d'administration éloignées à l'aide de WebSphere MQ Explorer, l'utilisateur exécutant WebSphere MQ Explorer doit disposer des droits suivants:

- Droits CONNECT sur l'objet gestionnaire de files d'attente cible
- Droits INQUIRE sur l'objet de gestionnaire de files d'attente cible
- Droits DISPLAY sur l'objet gestionnaire de files d'attente cible
- Droits INQUIRE sur la file d'attente, SYSTEM.MQEXPLORER.REPLY.MODEL
- Droits DISPLAY sur la file d'attente, SYSTEM.MQEXPLORER.REPLY.MODEL
- Droit d'entrée (get) sur la file d'attente, SYSTEM.MQEXPLORER.REPLY.MODEL
- Droit de sortie (put) sur la file d'attente, SYSTEM.ADMIN.COMMAND.QUEUE
- Droits INQUIRE sur la file d'attente, SYSTEM.ADMIN.COMMAND.QUEUE
- Droit d'exécution de l'action sélectionnée

Remarque : Les droits d'entrée sont liés à l'entrée à l'utilisateur à partir d'une file d'attente (opération d'extraction). Le droit OUTPUT est lié à la sortie de l'utilisateur vers une file d'attente (opération d'insertion).

Pour vous connecter à un gestionnaire de files d'attente éloignées sur WebSphere MQ for z/OS et effectuer des tâches d'administration à distance à l'aide de IBM WebSphere MQ Explorer, vous devez fournir les informations suivantes:

- Un profil RACF pour la file d'attente système, SYSTEM.MQEXPLORER.REPLY.MODEL
- Un profil RACF pour les files d'attente, AMQ.MQEXPLORER. *

En outre, l'utilisateur exécutant WebSphere MQ Explorer doit disposer des droits suivants:

- RACF Droits UPDATE sur la file d'attente système, SYSTEM.MQEXPLORER.REPLY.MODEL
- Droits de mise à jour RACF sur les files d'attente, AMQ.MQEXPLORER. *
- Droits CONNECT sur l'objet gestionnaire de files d'attente cible
- Droit d'exécution de l'action sélectionnée
- Droits d'accès en lecture (READ) à tous les profils hlq.DISPLAY.object dans la classe MQCMDS

Pour plus d'informations sur l'octroi de droits d'accès aux objets WebSphere MQ , voir Attribution de droits d'accès à un objet WebSphere MQ sur les systèmes UNIX ou Linux et Windows.

Si un utilisateur tente d'effectuer une opération qu'il n'est pas autorisé à effectuer, le gestionnaire de files d'attente cible appelle des procédures d'échec d'autorisation et l'opération échoue.

Le filtre par défaut dans WebSphere MQ Explorer consiste à afficher tous les objets WebSphere MQ . S'il existe des objets WebSphere MQ sur lesquels un utilisateur ne dispose pas des droits DISPLAY, des échecs d'autorisation sont générés. Si des événements de droits sont enregistrés, limitez la plage d'objets affichés aux objets sur lesquels l'utilisateur dispose des droits DISPLAY.

Sécurité pour la connexion aux gestionnaires de files d'attente éloignées

Vous devez sécuriser le canal entre IBM WebSphere MQ Explorer et chaque gestionnaire de files d'attente éloignées.

IBM WebSphere MQ Explorer se connecte aux gestionnaires de files d'attente éloignées en tant qu'application client MQI. Cela signifie que chaque gestionnaire de files d'attente éloignées doit disposer d'une définition de canal de connexion serveur et d'un programme d'écoute TCP/IP approprié. Si vous ne sécurisez pas votre canal de connexion serveur, une application malveillante peut se connecter au même canal de connexion serveur et accéder aux objets du gestionnaire de files d'attente avec des droits illimités. Pour sécuriser votre canal de connexion serveur, indiquez une valeur non vide pour l'attribut MCAUSER du canal, utilisez des enregistrements d'authentification de canal ou utilisez un exit de sécurité.

La valeur par défaut de l'attribut MCAUSER est l'ID utilisateur local. Si vous indiquez un nom d'utilisateur non vide comme attribut MCAUSER du canal de connexion serveur, tous les programmes qui se connectent au gestionnaire de files d'attente à l'aide de ce canal s'exécutent avec l'identité de l'utilisateur nommé et disposent du même niveau de droits d'accès. Cela ne se produit pas si vous utilisez des enregistrements d'authentification de canal.

Utilisation d'un exit de sécurité avec WebSphere MQ Explorer

Vous pouvez spécifier un exit de sécurité par défaut et des exits de sécurité spécifiques au gestionnaire de files d'attente à l'aide de l'explorateur WebSphere MQ .

Vous pouvez définir un exit de sécurité par défaut, qui peut être utilisé pour toutes les nouvelles connexions client à partir de WebSphere MQ Explorer. Cet exit par défaut peut être remplacé lors de l'établissement d'une connexion. Vous pouvez également définir un exit de sécurité pour un seul gestionnaire de files d'attente ou un ensemble de gestionnaires de files d'attente, qui prend effet lorsqu'une connexion est établie. Vous spécifiez des exits à l'aide de WebSphere MQ Explorer. Pour plus d'informations, voir le centre d'aide WebSphere MQ .

Utilisation de IBM WebSphere MQ Explorer pour la connexion à un gestionnaire de files d'attente éloignées à l'aide de canaux MQI compatibles SSL

IBM WebSphere MQ Explorer se connecte aux gestionnaires de files d'attente éloignées à l'aide d'un canal MQI. Si vous souhaitez sécuriser le canal MQI à l'aide de la sécurité SSL, vous devez établir le canal à l'aide d'une table de définition de canal du client.

Pour plus d'informations sur l'établissement d'un canal MQI à l'aide d'une table de définition de canal du client, voir Présentation des clients IBM WebSphere MQ MQI.

Une fois que vous avez établi le canal à l'aide d'une table de définition de canal du client, vous pouvez utiliser IBM WebSphere MQ Explorer pour vous connecter à un gestionnaire de files d'attente éloignées à l'aide d'un canal MQI SSL, comme décrit dans «Tâches sur le système qui héberge le gestionnaire de files d'attente éloignées», à la page 63 et «Tâches sur le système qui héberge le IBM WebSphere MQ Explorer», à la page 64.

Tâches sur le système qui héberge le gestionnaire de files d'attente éloignées

Sur le système hébergeant le gestionnaire de files d'attente éloignées, effectuez les tâches suivantes:

1. Définissez une paire de canaux de connexion serveur et de connexion client et spécifiez la valeur appropriée pour la variable *SSLCIPH* sur la connexion serveur sur les deux canaux. Pour plus d'informations sur la variable *SSLCIPH* , voir Protection des canaux avec SSL .
2. Envoyez la table de définition de canal `AMQCLCHL.TAB` , qui se trouve dans le répertoire `@ipcc` du gestionnaire de files d'attente, au système hébergeant IBM WebSphere MQ Explorer.
3. Démarrez un programme d'écoute TCP/IP sur un port désigné.
4. Placez les certificats SSL de l'autorité de certification et les certificats SSL personnels dans le répertoire SSL du gestionnaire de files d'attente:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` pour les systèmes UNIX and Linux
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` pour les systèmes Windows

   Où +QMNAME+ est un jeton représentant le nom du gestionnaire de files d'attente.
5. Créez un fichier de base de données de clés de type CMS nommé `key.kdb` . Stockez le mot de passe dans un fichier en vérifiant l'option dans l'interface graphique iKeyman ou en utilisant l'option `-stash` avec les commandes **runmqckm** .
6. Ajoutez les certificats de l'autorité de certification à la base de données de clés créée à l'étape précédente.
7. Importez le certificat personnel du gestionnaire de files d'attente dans la base de données de clés.

Pour plus d'informations sur l'utilisation de la couche Secure Sockets Layer sur les systèmes Windows , voir Utilisation de SSL ou TLS sur les systèmes UNIX, Linux et Windows .

Tâches sur le système qui héberge le IBM WebSphere MQ Explorer

Sur le système hébergeant le IBM WebSphere MQ Explorer, effectuez les tâches suivantes:

1. Créez un fichier de base de données de clés de type JKS nommé `key.jks`. Définissez un mot de passe pour ce fichier de base de données de clés.

   Le IBM WebSphere MQ Explorer utilise les fichiers de clés Java (JKS) pour la sécurité SSL. Par conséquent, le fichier de clés créé pour configurer SSL pour IBM WebSphere MQ Explorer doit correspondre.
2. Ajoutez les certificats de l'autorité de certification à la base de données de clés créée à l'étape précédente.
3. Importez le certificat personnel du gestionnaire de files d'attente dans la base de données de clés.
4. Sur les systèmes Windows et Linux , démarrez MQ Explorer à l'aide du menu système, du fichier exécutable `MQExplorer` ou de la commande **`strmqcfg`** .
5. Dans la barre d'outils IBM WebSphere MQ Explorer , cliquez sur **Fenêtre-> Préférences**, puis développez **WebSphere MQ Explorer** et cliquez sur **Magasins de certificats client SSL**. Entrez le nom et le mot de passe du fichier JKS créé à l'étape 1 de «Tâches sur le système qui héberge le IBM WebSphere MQ Explorer», à la page 64, dans le magasin de certificats de confiance et le magasin de certificats personnels, puis cliquez sur **OK**.
6. Fermez la fenêtre **Préférences** et cliquez avec le bouton droit de la souris sur **Gestionnaires de files d'attente**. Cliquez sur **Afficher / Masquer les gestionnaires de files d'attente**, puis sur **Ajouter** dans l'écran **Afficher / Masquer les gestionnaires de files d'attente** .
7. Entrez le nom du gestionnaire de files d'attente et sélectionnez l'option **Se connecter directement** . Cliquez sur Suivant.
8. Sélectionnez **Utiliser la table de définition de canal du client (CCDT)** et indiquez l'emplacement du fichier de table de canaux que vous avez transféré du gestionnaire de files d'attente éloignées à l'étape 2 dans «Tâches sur le système qui héberge le gestionnaire de files d'attente éloignées», à la page 63 sur le système hébergeant le gestionnaire de files d'attente éloignées.
9. Cliquez sur **Finir**. Vous pouvez maintenant accéder au gestionnaire de files d'attente éloignées à partir de la IBM WebSphere MQ Explorer.

Connexion via un autre gestionnaire de files d'attente

L'explorateur IBM WebSphere MQ vous permet de vous connecter à un gestionnaire de files d'attente via un gestionnaire de files d'attente intermédiaire auquel l'explorateur IBM WebSphere MQ est déjà connecté.

Dans ce cas, IBM WebSphere MQ Explorer insère des messages de commande PCF dans le gestionnaire de files d'attente intermédiaire, en spécifiant les éléments suivants:

- Le paramètre *ObjectQMgrNom* dans le descripteur d'objet (MQOD) comme nom du gestionnaire de files d'attente cible. Pour plus d'informations sur la résolution de nom de file d'attente, voir Résolution de nom .
- Le paramètre *UserIdentifier* dans le descripteur de message (MQMD) en tant que userIdlocal.

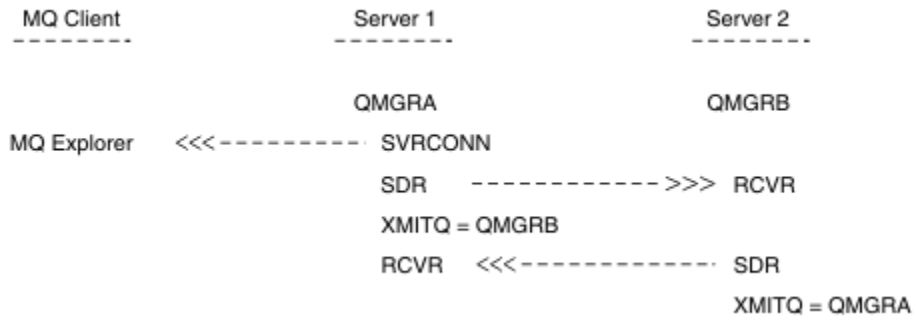
Si la connexion est ensuite utilisée pour se connecter au gestionnaire de files d'attente cible via un gestionnaire de files d'attente intermédiaire, l'ID utilisateur userId est de nouveau transmis dans le paramètre *UserIdentifier* du descripteur de message (MQMD). Pour que le programme d'écoute MCA sur le gestionnaire de files d'attente cible accepte ce message, l'attribut MCAUSER doit être défini ou l' userId doit déjà exister avec le droit d'insertion.

Le serveur de commandes du gestionnaire de files d'attente cible insère des messages dans la file d'attente de transmission en spécifiant userId dans le paramètre *UserIdentifier* du descripteur de message (MQMD). Pour que cette insertion aboutisse, l' userId doit déjà exister sur le gestionnaire de files d'attente cible avec les droits d'insertion.

L'exemple suivant montre comment connecter un gestionnaire de files d'attente, via un gestionnaire de files d'attente intermédiaire, à WebSphere MQ Explorer.

Etablissez une connexion d'administration à distance à un gestionnaire de files d'attente. Vérifiez que:

- Le gestionnaire de files d'attente sur le serveur est actif et un canal de connexion serveur (SVRCONN) est défini.
- Le programme d'écoute est actif.
- Le serveur de commandes est actif.
- SYSTEME SYSTEM.MQ EXPLORER.REPLY.MODEL a été créée et vous disposez des droits suffisants.
- Les programmes d'écoute du gestionnaire de files d'attente, les serveurs de commandes et les canaux émetteurs sont démarrés.



Dans cet exemple :

- IBM WebSphere MQ Explorer est connecté au gestionnaire de files d'attente QMGRA (exécuté sur Server1) à l'aide d'une connexion client.
- Le gestionnaire de files d'attente QMGRB sur Server2 peut désormais être connecté à IBM WebSphere MQ Explorer via un gestionnaire de files d'attente intermédiaire (QMGRA)
- Lors de la connexion à QMGRB avec WebSphere MQ Explorer, sélectionnez QMGRA comme gestionnaire de files d'attente intermédiaire

Dans cette situation, il n'y a pas de connexion directe à QMGRB à partir de IBM WebSphere MQ Explorer ; la connexion à QMGRB se fait via QMGRA.

Le gestionnaire de files d'attente QMGRB sur Server2 est connecté à QMGRA sur Server1 à l'aide de canaux émetteurs-récepteurs. Le canal entre QMGRA et QMGRB doit être configuré de sorte que l'administration à distance soit possible ; voir «Préparation des canaux et des files d'attente de transmission pour l'administration à distance», à la page 111.

Affichage et masquage des gestionnaires de files d'attente et des clusters

IBM WebSphere MQ Explorer peut afficher plusieurs gestionnaires de files d'attente à la fois. Dans le panneau Afficher / Masquer le gestionnaire de files d'attente (sélectionnable dans le menu du noeud d'arborescence des gestionnaires de files d'attente), vous pouvez choisir d'afficher ou non des informations sur une autre machine (distante). Les gestionnaires de files d'attente locaux sont détectés automatiquement.

Pour afficher un gestionnaire de files d'attente éloignées:

1. Cliquez avec le bouton droit de la souris sur le noeud d'arborescence `Queue Managers` , puis sélectionnez *Afficher / Masquer les gestionnaires de files d'attente ...*.
2. Cliquez sur **Ajouter**. Le panneau Afficher / Masquer les gestionnaires de files d'attente s'affiche.
3. Entrez le nom du gestionnaire de files d'attente éloignées et le nom d'hôte ou l'adresse IP dans les zones fournies.

   Le nom d'hôte ou l'adresse IP est utilisé pour établir une connexion client au gestionnaire de files d'attente éloignées à l'aide de son canal de connexion serveur par défaut, SYSTEM.ADMIN.SVRCONN, ou un canal de connexion serveur défini par l'utilisateur.
4. Cliquez sur **Terminer**.

Le panneau Afficher / Masquer les gestionnaires de files d'attente affiche également la liste de tous les gestionnaires de files d'attente visibles. Vous pouvez utiliser ce panneau pour masquer les gestionnaires de files d'attente dans la vue de navigation.

Si IBM WebSphere MQ Explorer affiche un gestionnaire de files d'attente qui est membre d'un cluster, le cluster est détecté et affiché automatiquement.

Pour exporter la liste des gestionnaires de files d'attente éloignées à partir de ce panneau:

1. Fermez le panneau Afficher / Masquer les gestionnaires de files d'attente.
2. Cliquez avec le bouton droit de la souris sur le noeud d'arborescence **IBM WebSphere MQ** supérieur dans la sous-fenêtre de navigation de l'explorateur WebSphere MQ , puis sélectionnez **Exporter les paramètres de l'explorateur MQ**
3. Cliquez sur **MQ Explorer > MQ Explorer Settings**
4. Sélectionnez **Informations de connexion > Gestionnaires de files d'attente éloignées**.
5. Sélectionnez un fichier dans lequel stocker les paramètres exportés.
6. Enfin, cliquez sur **Terminer** pour exporter les informations de connexion du gestionnaire de files d'attente éloignées dans le fichier spécifié.

Pour importer une liste de gestionnaires de files d'attente éloignées:

1. Cliquez avec le bouton droit de la souris sur le noeud d'arborescence **IBM WebSphere MQ** supérieur dans la sous-fenêtre de navigation de WebSphere MQ Explorer, puis sélectionnez **Importer les paramètres MQ Explorer**
2. Cliquez sur **MQ Explorer > MQ Explorer Settings**
3. Cliquez sur **Parcourir**et accédez au chemin du fichier contenant les informations de connexion du gestionnaire de files d'attente éloignées.
4. Cliquez sur **Ouvrir**. Si le fichier contient une liste de gestionnaires de files d'attente éloignées, la case **Informations de connexion > Gestionnaires de files d'attente éloignées** est cochée.
5. Enfin, cliquez sur **Terminer** pour importer les informations de connexion du gestionnaire de files d'attente éloignées dans WebSphere MQ Explorer.

Appartenance au(x) cluster(s)

IBM WebSphere MQ Explorer requiert des informations sur les gestionnaires de files d'attente qui sont membres d'un cluster.

Si un gestionnaire de files d'attente est membre d'un cluster, le noeud de l'arborescence de clusters est renseigné automatiquement.

Si des gestionnaires de files d'attente deviennent membres de clusters alors que IBM WebSphere MQ Explorer est en cours d'exécution, vous devez gérer IBM WebSphere MQ Explorer avec des données d'administration à jour sur les clusters afin qu'il puisse communiquer efficacement avec eux et afficher des informations de cluster correctes lorsque cela est demandé. Pour ce faire, WebSphere MQ Explorer a besoin des informations suivantes:

- Nom d'un gestionnaire de files d'attente de référentiel
- Nom de connexion du gestionnaire de files d'attente de référentiel s'il se trouve sur un gestionnaire de files d'attente éloignées

Avec ces informations, WebSphere MQ Explorer peut:

- Utilisez le gestionnaire de files d'attente de référentiel pour obtenir la liste des gestionnaires de files d'attente du cluster.
- Administrez les gestionnaires de files d'attente qui sont membres du cluster et qui se trouvent sur des plateformes et des niveaux de commande pris en charge.

L'administration n'est pas possible si:

- Le référentiel choisi devient indisponible. WebSphere MQ Explorer ne bascule pas automatiquement vers un autre référentiel.
- Le référentiel choisi ne peut pas être contacté via TCP/IP.
- Le référentiel choisi s'exécute sur un gestionnaire de files d'attente qui s'exécute sur une plateforme et un niveau de commande non pris en charge par WebSphere MQ Explorer.

Les membres du cluster qui peuvent être administrés peuvent être locaux ou distants s'ils peuvent être contactés à l'aide de TCP/IP. IBM WebSphere MQ Explorer se connecte aux gestionnaires de files d'attente locaux qui sont membres d'un cluster directement, sans utiliser de connexion client.

Conversion de données

IBM WebSphere MQ Explorer fonctionne avec le CCSID 1208 (UTF-8). Cela permet à IBM WebSphere MQ Explorer d'afficher correctement les données des gestionnaires de files d'attente éloignées. Que vous vous connectiez à un gestionnaire de files d'attente directement ou à l'aide d'un gestionnaire de files d'attente intermédiaire, IBM WebSphere MQ Explorer requiert la conversion de tous les messages entrants au CCSID 1208 (UTF-8).

Un message d'erreur est émis si vous tentez d'établir une connexion entre IBM WebSphere MQ Explorer et un gestionnaire de files d'attente avec un CCSID que IBM WebSphere MQ Explorer ne reconnaît pas.

Les conversions prises en charge sont décrites dans la rubrique Conversion de page de codes.

Sécurité sous Windows

L'assistant de préparation de WebSphere MQ crée un compte utilisateur spécial afin que le service Windows puisse être partagé par les processus qui doivent l'utiliser.

Un service Windows est partagé entre les processus client pour une installation IBM WebSphere MQ . Un service est créé pour chaque installation. Chaque service est nommé `MQ_`*`InstallationName`*et possède le nom d'affichage `IBM WebSphere MQ(`*`InstallationName`*`)`. Avant Version 7.1, avec une seule installation sur un serveur, le service Windows était nommé `MQSeriesServices` avec le nom d'affichage `IBM MQSeries`.

Etant donné que chaque service doit être partagé entre des sessions de connexion non interactives et interactives, vous devez le lancer sous un compte utilisateur spécial. Vous pouvez utiliser un compte utilisateur spécial pour tous les services ou créer des comptes utilisateur spéciaux différents. Chaque compte utilisateur spécial doit avoir le droit de "se connecter en tant que service". Pour plus d'informations, voir «Droits utilisateur requis pour un service IBM WebSphere MQ Windows», à la page 68. Si l'ID utilisateur ne dispose pas des droits pour exécuter le service, ce dernier ne démarre pas et renvoie une erreur dans le journal des événements du système Windows. En général, vous aurez exécuté l'assistant de préparation d'IBM WebSphere MQ et configuré l'ID utilisateur correctement. Toutefois, si vous avez configuré l'ID utilisateur manuellement, il se peut que vous ayez un problème à résoudre.

Lorsque vous installez IBM WebSphere MQ et exécutez l'assistant de préparation de IBM WebSphere MQ pour la première fois, il crée un compte utilisateur local pour le service appelé MUSR_MQADMIN avec les paramètres et les droits requis, y compris "Connexion en tant que service".

Pour les installations suivantes, l'assistant de préparation d' IBM WebSphere MQ crée un compte utilisateur nommé MUSR_MQADMINx, où x est le prochain numéro disponible représentant un ID utilisateur qui n'existe pas. Le mot de passe de MUSR_MQADMINx est généré de manière aléatoire lors de la création du compte et utilisé pour configurer l'environnement de connexion pour le service. Le mot de passe généré n'expire pas.

Ce compte IBM WebSphere MQ n'est affecté par aucune règle de compte configurée sur le système pour exiger que les mots de passe de compte soient modifiés après une certaine période.

Le mot de passe n'est pas connu en dehors de ce traitement à utilisation unique et est stocké par le système d'exploitation Windows dans une partie sécurisée du registre.

Utilisation d'Active Directory (Windows uniquement)

Dans certaines configurations réseau, où les comptes utilisateur sont définis sur les contrôleurs de domaine qui utilisent Active Directory, le compte utilisateur local IBM WebSphere MQ ne dispose peut-être pas des droits requis pour interroger l'appartenance à un groupe d'autres comptes utilisateur de domaine. L'assistant de préparation d' IBM WebSphere MQ identifie si tel est le cas en effectuant des tests et en posant des questions à l'utilisateur sur la configuration du réseau.

Si le compte utilisateur local sous lequel IBM WebSphere MQ s'exécute ne dispose pas des droits requis, l'assistant de préparation d' IBM WebSphere MQ demande à l'utilisateur les détails du compte d'un compte utilisateur de domaine avec des droits d'utilisateur particuliers. Pour connaître les droits utilisateur requis par le compte utilisateur de domaine, voir «Droits utilisateur requis pour un service IBM WebSphere MQ Windows», à la page 68. Une fois que l'utilisateur a entré des détails de compte valides pour le compte utilisateur de domaine dans l'assistant de préparation d' IBM WebSphere MQ , il configure un service IBM WebSphere MQ Windows à exécuter sous le nouveau compte. Les détails du compte sont conservés dans la partie sécurisée du registre et ne peuvent pas être lus par les utilisateurs.

Lorsque le service est en cours d'exécution, un service IBM WebSphere MQ Windows est lancé et reste en cours d'exécution tant que le service est en cours d'exécution. Un administrateur IBM WebSphere MQ qui se connecte au serveur après le lancement du service Windows peut utiliser le IBM WebSphere MQ Explorer pour administrer les gestionnaires de files d'attente sur le serveur. Le IBM WebSphere MQ Explorer est ainsi connecté au processus de service Windows existant. Ces deux actions nécessitent des niveaux d'autorisation différents pour pouvoir fonctionner:

- Le processus de lancement requiert un droit de lancement.
- L'administrateur IBM WebSphere MQ requiert des droits d'accès.

Droits utilisateur requis pour un service IBM WebSphere MQ Windows

Le tableau de cette rubrique répertorie les droits utilisateur requis pour le compte utilisateur local et de domaine sous lequel s'exécute le service Windows pour une installation IBM WebSphere MQ .

Se connecter en tant que travail par lots	Permet à un service IBM WebSphere MQ Windows de s'exécuter sous ce compte utilisateur.
Ouvrir une session en tant que service	Permet aux utilisateurs de définir le service IBM WebSphere MQ Windows pour se connecter à l'aide du compte configuré.
Arrêter le système	Permet au service IBM WebSphere MQ Windows de redémarrer le serveur s'il est configuré pour le faire en cas d'échec de la reprise d'un service.
Augmenter les quotas	Obligatoire pour l'appel `CreateProcessAsUser` du système d'exploitation.
Agir comme partie intégrante du système d'exploitation	Obligatoire pour l'appel `LogonUser` du système d'exploitation.
Ignorer le contrôle transversal	Obligatoire pour l'appel `LogonUser` du système d'exploitation.
Remplacer un jeton de processus	Obligatoire pour l'appel `LogonUser` du système d'exploitation.

Remarque : Les droits des programmes de débogage peuvent être nécessaires dans les environnements exécutant des applications ASP et IIS.

Votre compte utilisateur de domaine doit avoir ces droits d'utilisateur Windows définis en tant que droits d'utilisateur effectifs, comme indiqué dans l'application Stratégie de sécurité locale. Si ce n'est pas le cas, définissez-les à l'aide de l'application Stratégie de sécurité locale en local sur le serveur ou à l'aide du domaine d'application de sécurité du domaine.

Modification du nom d'utilisateur associé au service IBM WebSphere MQ

Vous devrez peut-être changer le nom d'utilisateur associé au service IBM WebSphere MQ de MUSR_MQADMIN à autre chose. (Par exemple, vous pouvez être amené à effectuer cette opération si votre gestionnaire de files d'attente est associé à DB2, qui n'accepte pas les noms d'utilisateur de plus de 8 caractères.)

Procédure

1. Créer un nouveau compte utilisateur (par exemple, **NEW_NAME**)
2. Utilisez l'assistant de préparation d' IBM WebSphere MQ pour entrer les détails du nouveau compte utilisateur.

Modification du mot de passe du compte utilisateur de service IBM WebSphere MQ Windows

Pourquoi et quand exécuter cette tâche

Pour modifier le mot de passe du compte utilisateur local du serviceIBM WebSphere MQ Windows , procédez comme suit:

Procédure

1. Identifiez l'utilisateur sous lequel le service s'exécute.
2. Arrêtez le service IBMIBM WebSphere MQ à partir du panneau Gestion de l'ordinateur.
3. Modifiez le mot de passe requis de la même manière que vous modifiez le mot de passe d'une personne.
4. Accédez aux propriétés du service IBM WebSphere MQ à partir du panneau Gestion de l'ordinateur.
5. Sélectionnez la page **Connexion** .
6. Vérifiez que le nom de compte spécifié correspond à l'utilisateur pour lequel le mot de passe a été modifié.
7. Entrez le mot de passe dans les zones **Mot de passe** et **Confirmer le mot de passe** , puis cliquez sur **OK**.

Service IBM WebSphere MQ Windows pour une installation s'exécutant sous un compte utilisateur de domaine

Pourquoi et quand exécuter cette tâche

Si le service IBM WebSphere MQ Windows d'une installation s'exécute sous un compte utilisateur de domaine, vous pouvez également modifier le mot de passe du compte comme suit:

Procédure

1. Modifiez le mot de passe du compte de domaine sur le contrôleur de domaine. Vous devrez peut-être demander à votre administrateur de domaine de le faire pour vous.
2. Suivez les étapes de modification de la page **Connexion** pour le service IBMIBM WebSphere MQ .

   Le compte utilisateur sous lequel s'exécute le service IBM WebSphere MQ Windows exécute toutes les commandes MQSC émises par les applications de l'interface utilisateur ou exécutées automatiquement lors du démarrage du système, de l'arrêt ou de la reprise du service. Ce compte utilisateur doit donc disposer des droits d'administration IBM WebSphere MQ . Par défaut, il est ajouté au groupe **mqm** local sur le serveur. Si cette appartenance est supprimée, le service IBM WebSphere MQ Windows ne fonctionne pas. Pour plus d'informations sur les droits utilisateur, voir «Droits utilisateur requis pour un service IBM WebSphere MQ Windows», à la page 68

Si un problème de sécurité survient avec le compte utilisateur sous lequel le service IBM WebSphere MQ Windows s'exécute, des messages d'erreur et des descriptions apparaissent dans le journal des événements système.

Concepts associés

«Utilisation d'Active Directory (Windows uniquement)», à la page 68
Dans certaines configurations réseau, où les comptes utilisateur sont définis sur les contrôleurs de domaine qui utilisent Active Directory, le compte utilisateur local IBM WebSphere MQ ne dispose peut-être pas des droits requis pour interroger l'appartenance à un groupe d'autres comptes utilisateur de domaine. L'assistant de préparation d' IBM WebSphere MQ identifie si tel est le cas en effectuant des tests et en posant des questions à l'utilisateur sur la configuration du réseau.

IBM WebSphere MQ coordination avec Db2 en tant que gestionnaire de ressources

Si vous démarrez vos gestionnaires de files d'attente à partir de IBM WebSphere MQ Explorerou si vous utilisez IBM WebSphere MQ V7et que vous rencontrez des problèmes lors de la coordination de Db2, consultez les journaux d'erreurs de votre gestionnaire de files d'attente.

Recherchez dans les journaux d'erreurs de votre gestionnaire de files d'attente une erreur telle que la suivante:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
    for xa_open. The queue manager is continuing without this resource manager.
```

Explication: L'ID utilisateur (nom par défaut: MUSR_MQADMIN) qui exécute leIBM WebSphere MQ processus de service `amqsvc.exe` est toujours en cours d'exécution avec un jeton d'accès qui ne contient pas d'informations d'appartenance au groupe pour le groupe DB2USERS.

Résoudre: Après avoir vérifié que l'ID utilisateur du service IBM WebSphere MQ est membre de DB2USERS, utilisez la séquence de commandes suivante:

- Arrêtez le service.
- arrêter tous les autres processus exécutés sous le même ID utilisateur.
- redémarrer ces processus.

Le réamorçage de la machine permet de vérifier les étapes précédentes, mais n'est pas nécessaire.

Extension du IBM WebSphere MQ Explorer

IBM WebSphere MQ for Windowset IBM WebSphere MQ for Linux (plateformesx86 et x86-64) fournissent une interface d'administration appelée IBM WebSphere MQ Explorer pour effectuer des tâches d'administration au lieu d'utiliser des commandes de contrôle ou MQSC.

Ces informations s'appliquent à WebSphere MQ for Windows et WebSphere MQ for Linux (plateformesx86 et x86-64) uniquement.

L'explorateur IBM WebSphere MQ présente les informations dans un style cohérent avec celui de l'infrastructure Eclipse et des autres applications de plug-in prises en charge par Eclipse .

Grâce à l'extension de IBM WebSphere MQ Explorer, les administrateurs système ont la possibilité de personnaliser WebSphere MQ Explorer afin d'améliorer la façon dont ils administrent WebSphere MQ.

Pour plus d'informations, voir *Extension de l'explorateur IBM WebSphere MQ* dans la documentation du produit IBM WebSphere MQ Explorer.

Utilisation de l'application IBM WebSphere MQ Taskbar (Windows uniquement)

L'application IBM WebSphere MQ Taskbar affiche une icône dans la barre des tâches système Windows sur le serveur. L'icône vous fournit le statut en cours de IBM WebSphere MQ et un menu à partir duquel vous pouvez effectuer des actions simples.

Sous Windows, l'icône WebSphere MQ se trouve dans la barre des tâches système du serveur et est remplacée par un symbole de statut codé par couleur, qui peut avoir l'une des significations suivantes:

Vert
Fonctionnement correct ; aucune alerte à l'heure actuelle

Bleu
Indéterminé ; WebSphere MQ est en cours de démarrage ou d'arrêt

Jaune
Alerte ; un ou plusieurs services échouent ou ont déjà échoué

Pour afficher le menu, cliquez avec le bouton droit de la souris sur l'icône WebSphere MQ . Dans le menu, vous pouvez effectuer les actions suivantes:

- Cliquez sur **Ouvrir** pour ouvrir le moniteur d'alertes WebSphere MQ
- Cliquez sur **Quitter** pour quitter l'application WebSphere MQ Taskbar
- Cliquez sur **WebSphere MQ Explorer** pour démarrer l'explorateur IBM WebSphere MQ .
- Cliquez sur **Arrêter WebSphere MQ** pour arrêter WebSphere MQ
- Cliquez sur **A propos de WebSphere MQ** pour afficher des informations sur le moniteur d'alertes WebSphere MQ

L'application du moniteur d'alertes IBM WebSphere MQ (Windows uniquement)

Le moniteur d'alertes IBM WebSphere MQ est un outil de détection des erreurs qui identifie et enregistre les problèmes liés à IBM WebSphere MQ sur une machine locale.

Le moniteur d'alertes affiche des informations sur le statut en cours de l'installation locale d'un serveur WebSphere MQ . Il surveille également l'interface ACPI (Windows Advanced Configuration and Power Interface) et s'assure que les paramètres ACPI sont appliqués.

A partir du moniteur d'alertes WebSphere MQ , vous pouvez:

- Accédez directement à l'explorateur IBM WebSphere MQ
- Afficher les informations relatives à toutes les alertes en attente
- Arrêtez le service WebSphere MQ sur la machine locale
- Acheminer les messages d'alerte sur le réseau vers un compte utilisateur configurable ou vers un poste de travail ou un serveur Windows

Administration des objets IBM WebSphere MQ locaux

Cette section explique comment administrer les objets IBM WebSphere MQ locaux pour prendre en charge les programmes d'application qui utilisent l'interface MQI (Message Queue Interface). Dans ce contexte, l'administration locale implique la création, l'affichage, la modification, la copie et la suppression d'objets IBM WebSphere MQ .

Outre les approches détaillées dans cette section, vous pouvez utiliser l'explorateur IBM WebSphere MQ pour administrer des objets WebSphere MQ locaux ; voir «Administration à l'aide de IBM WebSphere MQ Explorer», à la page 58.

Cette section comporte les informations suivantes :

- Programmes d'application utilisant l'interface MQI

- «Exécution de tâches d'administration locales à l'aide des commandes MQSC», à la page 76
- «Utilisation des gestionnaires de files d'attente», à la page 85
- «Utilisation des files d'attente locales», à la page 86
- «Utilisation de files d'attente alias», à la page 91
- «Utilisation des files d'attente modèles», à la page 93
- «Utilisation des services», à la page 100
- «Gestion des objets pour le déclenchement», à la page 106

Démarrage et arrêt d'un gestionnaire de files d'attente

Cette rubrique présente l'arrêt et le démarrage d'un gestionnaire de files d'attente.

Démarrage d'un gestionnaire de files d'attente

Pour démarrer un gestionnaire de files d'attente, utilisez la commande `strmqm` comme suit:

```
strmqm saturn.queue.manager
```

Sur les systèmes WebSphere MQ for Windows et WebSphere MQ for Linux (plateformesx86 et x86-64), vous pouvez démarrer un gestionnaire de files d'attente comme suit:

1. Ouvrez l'explorateur IBM WebSphere MQ .
2. Sélectionnez le gestionnaire de files d'attente dans la vue Navigator .
3. Cliquez sur `Start`. Le gestionnaire de files d'attente démarre.

Si le démarrage du gestionnaire de files d'attente prend plus de quelques secondes, WebSphere MQ émet des messages d'information détaillant par intermittence la progression du démarrage.

La commande `strmqm` ne renvoie pas le contrôle tant que le gestionnaire de files d'attente n'a pas démarré et n'est pas prêt à accepter les demandes de connexion.

Démarrage automatique d'un gestionnaire de files d'attente

Dans WebSphere MQ for Windows , vous pouvez démarrer automatiquement un gestionnaire de files d'attente lorsque le système démarre à l'aide de WebSphere MQ Explorer. Pour plus d'informations, voir «Administration à l'aide de IBM WebSphere MQ Explorer», à la page 58.

Arrêt d'un gestionnaire de files d'attente

La commande **endmqm** permet d'arrêter un gestionnaire de files d'attente.

Remarque : Vous devez utiliser la commande **endmqm** à partir de l'installation associée au gestionnaire de files d'attente que vous utilisez. L'installation à laquelle un gestionnaire de files d'attente est associé peut être identifiée à l'aide de la commande `dspmq -o installation`.

Par exemple, pour arrêter un gestionnaire de files d'attente appelé `QMB`, entrez la commande suivante:

```
endmqm QMB
```

Sur les systèmes WebSphere MQ for Windows et WebSphere MQ for Linux (plateformesx86 et x86-64), vous pouvez arrêter un gestionnaire de files d'attente comme suit:

1. Ouvrez l'explorateur IBM WebSphere MQ .
2. Sélectionnez le gestionnaire de files d'attente dans la vue Navigator .
3. Cliquez sur `Stop...`. Le panneau Arrêt du gestionnaire de files d'attente s'affiche.
4. Sélectionnez Contrôlé ou Immédiat.
5. Cliquez sur OK. Le gestionnaire de files d'attente s'arrête.

arrêt progressif

Par défaut, la commande **endmqm** effectue un arrêt au repos du gestionnaire de files d'attente spécifié. Cette opération peut prendre un certain temps. Un arrêt mis au repos attend que toutes les applications connectées soient déconnectées.

Utilisez ce type d'arrêt pour avertir les applications de l'arrêt. Si vous émettez:

```
endmqm -c QMB
```

vous n'êtes pas informé de l'arrêt de toutes les applications. (Une commande `endmqm -c QMB` est équivalente à une commande `endmqm QMB` .)

Toutefois, si vous émettez:

```
endmqm -w QMB
```

la commande attend que toutes les applications soient arrêtées et que le gestionnaire de files d'attente soit arrêté.

Arrêt immédiat

Pour un arrêt immédiat, tous les appels MQI en cours sont autorisés à se terminer, mais tous les nouveaux appels échouent. Ce type d'arrêt n'attend pas que les applications se déconnectent du gestionnaire de files d'attente.

Pour un arrêt immédiat, entrez:

```
endmqm -i QMB
```

Arrêt préemptif

Remarque : N'utilisez cette méthode que si toutes les autres tentatives d'arrêt du gestionnaire de files d'attente à l'aide de la commande **endmqm** ont échoué. Cette méthode peut avoir des conséquences imprévisibles pour les applications connectées.

Si un arrêt immédiat ne fonctionne pas, vous devez recourir à un arrêt *préventif* en spécifiant l'indicateur `-p` . Exemple :

```
endmqm -p QMB
```

Le gestionnaire de files d'attente est arrêté immédiatement. Si cette méthode ne fonctionne toujours pas, voir «Arrêt manuel d'un gestionnaire de files d'attente», à la page 74 pour une autre solution.

Pour une description détaillée de la commande **endmqm** et de ses options, voir endmqm.

Si vous rencontrez des problèmes lors de l'arrêt d'un gestionnaire de files d'attente

Les problèmes liés à l'arrêt d'un gestionnaire de files d'attente sont souvent causés par des applications. Par exemple, lorsque des applications:

- Ne pas vérifier correctement les codes retour MQI
- Ne pas demander de notification de mise au repos
- Arrêter sans se déconnecter du gestionnaire de files d'attente (en émettant un appel `MQDISC`)

Si un problème se produit lorsque vous arrêtez le gestionnaire de files d'attente, vous pouvez sortir de la commande **endmqm** en utilisant Ctrl-C. Vous pouvez ensuite exécuter une autre commande **endmqm** , mais cette fois avec un indicateur qui spécifie le type d'arrêt dont vous avez besoin.

Arrêt manuel d'un gestionnaire de files d'attente

Si les méthodes standard d'arrêt des gestionnaires de files d'attente échouent, essayez les méthodes décrites ici.

La méthode standard d'arrêt des gestionnaires de files d'attente consiste à utiliser la commande `endmqm` . Pour arrêter un gestionnaire de files d'attente manuellement, utilisez l'une des procédures décrites dans cette section. Pour plus d'informations sur l'exécution d'opérations sur les gestionnaires de files d'attente à l'aide de commandes de contrôle, voir Création et gestion de gestionnaires de files d'attente.

Arrêt des gestionnaires de files d'attente sous Windows

Comment arrêter les processus et le service IBM WebSphere MQ pour arrêter les gestionnaires de files d'attente dans IBM WebSphere MQ for Windows.

Pour arrêter un gestionnaire de files d'attente exécuté sous WebSphere MQ for Windows:

1. Répertoriez les noms (ID) des processus en cours d'exécution à l'aide du gestionnaire de tâches Windows .
2. Arrêtez les processus à l'aide du gestionnaire de tâches Windows ou de la commande **taskkill** , dans l'ordre suivant (s'ils sont en cours d'exécution):

AMQZMUC0	Gestionnaire de processus critique
AMQZXMA0	Contrôleur d'exécution
AMQZFUMA	Processus OAM
AMQZLAA0	Agents LQM
AMQZLSA0	Agents LQM
AMQZMUF0	Gestionnaire d'utilitaire
AMQZMGR0	Ctrlur de processus
AMQZMUR0	Gestionnaire de processus redémarrable
AMQFQPUB	Processus de publication / d'abonnement
AMQFCXBA	Processus de travail de courtier
AMQRMPPA	Processus de regroupement de processus
amqcrsta	Processus de travail de répondeur non à unités d'exécution
AMQCRS6B	Connexion client et canal récepteur LU62
AMQRRMFA	Processus de référentiel (pour les clusters)
AMQZDMAA	Processeur de messages différés
AMQPCSEA	Serveur de commandes
runmqtrm	Appel d'un moniteur de déclenchement pour un serveur
runmqdlq	Appeler le gestionnaire de files d'attente de rebut
runmqchi	Processus de l'initiateur de canal
runmqlsr	Processus du programme d'écoute de canal
AMQXSSVN	Serveurs de mémoire partagée
AMQZTRCN	Fonction de trace

3. Arrêtez le service WebSphere MQ depuis **Outils d'administration** > **Services** dans le panneau de configuration Windows .
4. Si vous avez essayé toutes les méthodes et que le gestionnaire de files d'attente ne s'est pas arrêté, réamorcez votre système.

Le gestionnaire de tâches Windows et la commande **tasklist** fournissent des informations limitées sur les tâches. Pour plus d'informations permettant de déterminer quels processus sont liés à un gestionnaire de files d'attente particulier, utilisez un outil tel que *Process Explorer* (procexp.exe), disponible au téléchargement à partir du site Web de Microsoft à l'adresse https://www.microsoft.com.

Arrêt des gestionnaires de files d'attente sur les systèmes UNIX and Linux

Comment arrêter les processus et le service IBM WebSphere MQ pour arrêter les gestionnaires de files d'attente dans IBM WebSphere MQ for UNIX and Linux. Vous pouvez essayer les méthodes décrites ici si les méthodes standard d'arrêt et de suppression des gestionnaires de files d'attente échouent.

Pour arrêter un gestionnaire de files d'attente exécuté sous WebSphere MQ pour les systèmes UNIX and Linux :

1. Recherchez les ID de processus des programmes de gestionnaire de files d'attente qui sont toujours en cours d'exécution à l'aide de la commande ps . Par exemple, si le gestionnaire de files d'attente est appelé QMNAME, utilisez la commande suivante:

   ```
   ps -ef | grep QMNAME
   ```

2. Arrêtez les processus de gestionnaire de files d'attente qui sont toujours en cours d'exécution. Utilisez la commande kill en spécifiant les ID de processus reconnus à l'aide de la commande ps .

   Arrêtez les processus dans l'ordre suivant:

amqzmuc0	Gestionnaire de processus critique
amqzxma0	Contrôleur d'exécution
amqzfuma	Processus OAM
amqzlaa0	Agents LQM
amqzlsa0	Agents LQM
amqzmuf0	Gestionnaire d'utilitaire
amqzmur0	Gestionnaire de processus redémarrable
amqzmgr0	Ctrlur de processus
amqfqpub	Processus de publication / d'abonnement
amqfcxba	Processus de travail de courtier
amqrmppa	Processus de regroupement de processus
amqcrsta	Processus de travail de répondeur non à unités d'exécution
amqcrs6b	Connexion client et canal récepteur LU62
amqrrmfa	Processus de référentiel (pour les clusters)
amqzdmaa	Processeur de messages différés
amqpcsea	Serveur de commandes
runmqtrm	Appel d'un moniteur de déclenchement pour un serveur
runmqdlq	Appeler le gestionnaire de files d'attente de rebut
runmqchi	Processus de l'initiateur de canal
runmqlsr	Processus du programme d'écoute de canal

Remarque : Vous pouvez utiliser la commande **kill -9** pour arrêter les processus dont l'arrêt a échoué.

Si vous arrêtez le gestionnaire de files d'attente manuellement, des fichiers FFST peuvent être utilisés et les fichiers FDC placés dans /var/mqm/errors. ne considèrent pas cela comme un incident dans le gestionnaire de files d'attente.

Le gestionnaire de files d'attente redémarre normalement, même après que vous l'ayez arrêté à l'aide de cette méthode.

Exécution de tâches d'administration locales à l'aide des commandes MQSC

Cette section présente les commandes MQSC et explique comment les utiliser pour certaines tâches courantes.

Si vous utilisez IBM WebSphere MQ pour Windows ou IBM WebSphere MQ pour Linux (plateformesx86 et x86-64), vous pouvez également effectuer les opérations décrites dans cette section à l'aide de l'explorateur IBM WebSphere MQ . Pour plus d'informations, voir «Administration à l'aide de IBM WebSphere MQ Explorer», à la page 58.

Vous pouvez utiliser les commandes MQSC pour gérer les objets de gestionnaire de files d'attente, notamment le gestionnaire de files d'attente lui-même, les files d'attente, les définitions de processus, les canaux, les canaux de connexion client, les programmes d'écoute, les services, les listes de noms, les clusters et les objets d'informations d'authentification. Cette section traite des gestionnaires de files d'attente, des files d'attente et des définitions de processus. Pour plus d'informations sur l'administration du canal, du canal de connexion client et des objets programme d'écoute, voir Objets . Pour plus d'informations sur toutes les commandes MQSC de gestion des objets de gestionnaire de files d'attente, voir «Commandes de script (MQSC)», à la page 76.

Vous émettez des commandes MQSC vers un gestionnaire de files d'attente à l'aide de la commande `runmqsc` . (Pour plus de détails sur cette commande, voir runmqsc.) Vous pouvez effectuer cette opération de manière interactive, en émettant des commandes à partir d'un clavier, ou vous pouvez rediriger l'unité d'entrée standard (`stdin`) pour exécuter une séquence de commandes à partir d'un fichier texte ASCII. Dans les deux cas, le format des commandes est identique. (Pour plus d'informations sur l'exécution des commandes à partir d'un fichier texte, voir «Exécution de commandes MQSC à partir de fichiers texte», à la page 80.)

Vous pouvez exécuter la commande `runmqsc` de trois manières, en fonction des indicateurs définis dans la commande:

- Vérifiez une commande sans l'exécuter, où les commandes MQSC sont vérifiées sur un gestionnaire de files d'attente local, mais ne sont pas exécutées.
- Exécutez une commande sur un gestionnaire de files d'attente local, où les commandes MQSC sont exécutées sur un gestionnaire de files d'attente local.
- Exécutez une commande sur un gestionnaire de files d'attente éloignées, où les commandes MQSC sont exécutées sur un gestionnaire de files d'attente éloignées.

Vous pouvez également exécuter la commande suivie d'un point d'interrogation pour afficher la syntaxe.

Les attributs d'objet spécifiés dans les commandes MQSC sont affichés dans cette section en majuscules (par exemple, RQMNAME), bien qu'ils ne soient pas sensibles à la casse. Les noms d'attribut de commande MQSC sont limités à huit caractères. Les commandes MQSC sont disponibles sur d'autres plateformes, notamment IBM i et z/OS.

Les commandes MQSC sont récapitulées dans la collection de rubriques de la section MQSC reference .

Commandes de script (MQSC)

Les commandes MQSC fournissent une méthode uniforme d'émission de commandes lisibles par l'utilisateur sur les plateformes WebSphere MQ . Pour plus d'informations sur les commandes PCF (*programmable command format*), voir «Présentation des commandes PCF (Programmable Command Formats)», à la page 9.

Le format général des commandes est présenté dans Les commandes MQSC.

Vous devez respecter les règles suivantes lorsque vous utilisez des commandes MQSC:

- Chaque commande commence par un paramètre primaire (un verbe), suivi d'un paramètre secondaire (un nom). Il est ensuite suivi du nom ou du nom générique de l'objet (entre parenthèses) s'il en

existe un, qui figure dans la plupart des commandes. Ensuite, les paramètres peuvent généralement se produire dans n'importe quel ordre ; si un paramètre a une valeur correspondante, la valeur doit se produire directement après le paramètre auquel il se rapporte.

- Les mots clés, les parenthèses et les valeurs peuvent être séparés par un nombre quelconque de blancs et de virgules. Une virgule affichée dans les diagrammes de syntaxe peut toujours être remplacée par un ou plusieurs blancs. Au moins un blanc doit précéder immédiatement chaque paramètre (après le paramètre principal).
- N'importe quel nombre de blancs peut apparaître au début ou à la fin de la commande et entre les paramètres, la ponctuation et les valeurs. Par exemple, la commande suivante est valide:

```
ALTER QLOCAL  ('Account'  )           TRIGDPTH  (  1)
```

  Les blancs à l'intérieur d'une paire de guillemets sont significatifs.
- Des virgules supplémentaires peuvent apparaître partout où des blancs sont autorisés et sont traitées comme s'il s'agissait de blancs (à moins, bien sûr, qu'elles se trouvent dans des chaînes entre guillemets).
- Les paramètres répétés ne sont pas autorisés. La répétition d'un paramètre avec sa version "NO", comme dans REPLACE NOREPLACE, n'est pas non plus autorisée.
- Chaînes contenant des blancs, des caractères minuscules ou des caractères spéciaux autres que:
  - le point (.)
  - la barre oblique (/)
  - le caractère de soulignement (_)
  - le symbole de pourcentage (%)

  doivent être placés entre guillemets simples, sauf s'ils sont:
  - Les valeurs génériques se terminant par un astérisque
  - Un astérisque unique (par exemple, TRACE (*))
  - Une spécification de plage contenant un signe deux-points (par exemple, CLASS (01:03))

  Si la chaîne elle-même contient une apostrophe, celle-ci est représentée par deux apostrophes.Les caractères minuscules qui ne figurent pas entre guillemets sont convertis en majuscules.
- Sur les plateformes autres que z/OS, une chaîne ne contenant aucun caractère (c'est-à-dire deux guillemets simples sans espace entre eux) est interprétée comme un espace vide entre guillemets simples, c'est-à-dire interprétée de la même manière que (''). La seule exception à cette règle est si l'attribut utilisé est l'un des suivants:
  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR

  Deux guillemets simples sans espace sont interprétés comme une chaîne de longueur nulle.
- Dans v7.0, les blancs de fin des attributs de chaîne basés sur les types MQCHARV, tels que SELECTOR, les données de sous-utilisateur, sont traités comme significatifs, ce qui signifie que'abc' n'est pas égal à'abc'.
- Une parenthèse gauche suivie d'une parenthèse droite, sans aucune information significative entre les deux, par exemple

```
NAME ( )
```

  n'est pas valide, sauf indication contraire.

- Les mots clés ne sont pas sensibles à la casse: AltER, alter et ALTER sont tous acceptables.Tout ce qui n'est pas entre guillemets est plié en majuscules.
- Des synonymes sont définis pour certains paramètres.Par exemple, DEF est toujours un synonyme de DEFINE, de sorte que DEF QLOCAL est valide. Cependant, les synonymes ne sont pas uniquement des chaînes minimales ; DEFI n'est pas un synonyme valide de DEFINE.

  Remarque : Il n'existe aucun synonyme pour le paramètre DELETE. Cela permet d'éviter la suppression accidentelle d'objets lors de l'utilisation de DEF, synonyme de DEFINE.

Pour une présentation de l'utilisation des commandes MQSC pour l'administration de IBM WebSphere MQ, voir «Exécution de tâches d'administration locales à l'aide des commandes MQSC», à la page 76.

Les commandes MQSC utilisent certains caractères spéciaux pour avoir certaines significations. Pour plus d'informations sur ces caractères spéciaux et sur leur utilisation, voir Caractères ayant une signification particulière.

Pour savoir comment générer des scripts à l'aide de commandes MQSC, voir Génération de scripts de commande.

Pour la liste complète des commandes MQSC, voir Les commandes MQSC.

Tâches associées
Génération de scripts de commande

Noms d'objet WebSphere MQ

Comment utiliser les noms d'objet dans les commandes MQSC.

Dans les exemples, nous utilisons des noms longs pour les objets. Il s'agit de vous aider à identifier le type d'objet que vous traitez.

Lorsque vous émettez des commandes MQSC, vous devez spécifier uniquement le nom local de la file d'attente. Dans nos exemples, nous utilisons des noms de file d'attente tels que:

```
ORANGE.LOCAL.QUEUE
```

La partie `LOCAL.QUEUE` du nom illustre le fait que cette file d'attente est une file d'attente locale. Il n'est ***pas*** requis pour les noms des files d'attente locales en général.

Nous utilisons également le nom `saturn.queue.manager` comme nom de gestionnaire de files d'attente. La partie `queue.manager` du nom illustre le fait que cet objet est un gestionnaire de files d'attente. Il n'est *pas* requis pour les noms des gestionnaires de files d'attente en général.

Respect de la casse dans les commandes MQSC

Les commandes MQSC, y compris leurs attributs, peuvent être écrites en majuscules ou en minuscules. Les noms d'objet dans les commandes MQSC sont convertis en majuscules (c'est-à-dire que les files d'attente et les files d'attente ne sont pas différenciées), sauf si les noms sont placés entre apostrophes. Si des guillemets ne sont pas utilisés, l'objet est traité avec un nom en majuscules. Pour plus d'informations, voirRéférence MQSC .

L'appel de commande `runmqsc` , tout comme toutes les commandes de contrôle WebSphere MQ , est sensible à la casse dans certains environnements WebSphere MQ . Pour plus d'informations, voir Utilisation des commandes de contrôle .

Entrée et sortie standard

L' *unité d'entrée standard*, également appelée `stdin`, est l'unité à partir de laquelle l'entrée dans le système est effectuée. Il s'agit généralement du clavier, mais vous pouvez spécifier que l'entrée doit provenir d'un port série ou d'un fichier disque, par exemple. L' *unité de sortie standard*, également appelée `stdout`, est l'unité à laquelle la sortie du système est envoyée. Il s'agit généralement d'un affichage, mais vous pouvez rediriger la sortie vers un port série ou un fichier.

Sur les commandes du système d'exploitation et les commandes de contrôle WebSphere MQ , l'opérateur < redirige l'entrée. Si cet opérateur est suivi d'un nom de fichier, l'entrée est extraite du fichier. De même, l'opérateur > redirige la sortie ; si cet opérateur est suivi d'un nom de fichier, la sortie est dirigée vers ce fichier.

Utilisation des commandes MQSC en mode interactif

Vous pouvez utiliser les commandes MQSC de manière interactive à l'aide d'une fenêtre de commande ou d'un interpréteur de commandes.

Pour utiliser les commandes MQSC de manière interactive, ouvrez une fenêtre de commande ou un shell et entrez:

```
runmqsc
```

Dans cette commande, aucun nom de gestionnaire de files d'attente n'a été spécifié, de sorte que les commandes MQSC sont traitées par le gestionnaire de files d'attente par défaut. Si vous souhaitez utiliser un gestionnaire de files d'attente différent, indiquez le nom du gestionnaire de files d'attente dans la commande **runmqsc** . Par exemple, pour exécuter des commandes MQSC sur le gestionnaire de files d'attente `jupiter.queue.manager`, utilisez la commande suivante:

```
runmqsc jupiter.queue.manager
```

Après cela, toutes les commandes MQSC que vous entrez sont traitées par ce gestionnaire de files d'attente, en supposant qu'il se trouve sur le même noeud et qu'il est déjà en cours d'exécution.

Vous pouvez maintenant entrer n'importe quelle commande MQSC, selon les besoins. Par exemple, essayez celui-ci:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

Pour les commandes comportant trop de paramètres pour tenir sur une seule ligne, utilisez des caractères de continuation pour indiquer qu'une commande se poursuit sur la ligne suivante:

- Le signe moins (-) indique que la commande doit être poursuivie à partir du début de la ligne suivante.
- Un signe plus (+) indique que la commande doit être poursuivie à partir du premier caractère non blanc sur la ligne suivante.

L'entrée de commande se termine par le caractère final d'une ligne non vide qui n'est pas un caractère de continuation. Vous pouvez également arrêter l'entrée de commande de manière explicite en entrant un point-virgule (;). (Cela est particulièrement utile si vous entrez accidentellement un caractère de continuation à la fin de la dernière ligne de l'entrée de commande.)

Commentaires en retour des commandes MQSC

Lorsque vous émettez des commandes MQSC, le gestionnaire de files d'attente renvoie des messages d'opérateur qui confirment vos actions ou vous indiquent les erreurs que vous avez commises. Exemple :

```
AMQ8006: WebSphere MQ queue created.
```

Ce message confirme qu'une file d'attente a été créée.

```
AMQ8405: Syntax error detected at or near end of command segment below:-

AMQ8426: Valid MQSC commands are:


    ALTER
    CLEAR
```

```
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
     4 : end
```

Ce message indique que vous avez fait une erreur de syntaxe.

Ces messages sont envoyés au périphérique de sortie standard. Si vous n'avez pas entré la commande correctement, reportez-vous à la section Référence MQSC pour connaître la syntaxe correcte.

Arrêt de l'entrée interactive des commandes MQSC

Pour arrêter l'utilisation des commandes MQSC, entrez la commande END.

Vous pouvez également utiliser le caractère EOF pour votre système d'exploitation.

Exécution de commandes MQSC à partir de fichiers texte

L'exécution interactive de commandes MQSC convient aux tests rapides, mais si vous disposez de commandes très longues ou que vous utilisez une séquence de commandes particulière à plusieurs reprises, envisagez de rediriger `stdin` à partir d'un fichier texte.

«Entrée et sortie standard», à la page 78 contient des informations sur `stdin` et `stdout`. Pour rediriger `stdin` à partir d'un fichier texte, créez d'abord un fichier texte contenant les commandes MQSC à l'aide de votre éditeur de texte habituel. Lorsque vous utilisez la commande `runmqsc` , utilisez les opérateurs de redirection. Par exemple, la commande suivante exécute une séquence de commandes contenues dans le fichier texte `myprog.in`:

```
runmqsc < myprog.in
```

De même, vous pouvez également rediriger la sortie vers un fichier. Un fichier contenant les commandes MQSC pour l'entrée est appelé *fichier de commandes MQSC*. Le fichier de sortie contenant les réponses du gestionnaire de files d'attente est appelé *fichier de sortie*.

Pour rediriger à la fois `stdin` et `stdout` dans la commande `runmqsc` , utilisez la forme suivante de la commande:

```
runmqsc < myprog.in > myprog.out
```

Cette commande appelle les commandes MQSC contenues dans le fichier de commandes MQSC `myprog.in.` . Etant donné que nous n'avons pas spécifié de nom de gestionnaire de files d'attente, les commandes MQSC s'exécutent sur le gestionnaire de files d'attente par défaut. La sortie est envoyée au fichier texte `myprog.out`. Figure 12, à la page 81 montre un extrait du fichier de commandes MQSC `myprog.in` et Figure 13, à la page 82 montre l'extrait correspondant de la sortie dans `myprog.out`.

Pour rediriger `stdin` et `stdout` dans la commande `runmqsc` , pour un gestionnaire de files d'attente (`saturn.queue.manager`) qui n'est pas la valeur par défaut, utilisez la commande suivante:

```
runmqsc saturn.queue.manager < myprog.in > myprog.out
```

Fichiers de commandes MQSC

Les commandes MQSC sont écrites dans un format lisible par l'utilisateur, c'est-à-dire en texte ASCII. Figure 12, à la page 81 est un extrait d'un fichier de commandes MQSC affichant une commande MQSC

(DEFINE QLOCAL) avec ses attributs. Le Référence MQSC contient une description de chaque commande MQSC et sa syntaxe.

```
  .
  .
  .
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE  +
       DESCR(' ') +
       PUT(ENABLED) +
       DEFPRTY(0) +
       DEFPSIST(NO) +
       GET(ENABLED) +
       MAXDEPTH(5000) +
       MAXMSGL(1024) +
       DEFSOPT(SHARED) +
       NOHARDENBO +
       USAGE(NORMAL) +
       NOTRIGGER;
  .
  .
  .
```

Figure 12. Extraction à partir d'un fichier de commandes MQSC

Pour la portabilité entre les environnements WebSphere MQ , limitez la longueur de ligne dans les fichiers de commandes MQSC à 72 caractères. Le signe plus indique que la commande se poursuit sur la ligne suivante.

Rapports sur les commandes MQSC

La commande **runmqsc** renvoie un *rapport*, qui est envoyé à `stdout`. Le rapport contient:

- En-tête identifiant les commandes MQSC comme source du rapport:

  ```
  Starting MQSC for queue manager jupiter.queue.manager.
  ```

  Où `jupiter.queue.manager` est le nom du gestionnaire de files d'attente.
- Liste numérotée facultative des commandes MQSC émises. Par défaut, le texte de l'entrée est répercuté dans la sortie. Dans cette sortie, chaque commande est préfixée par un numéro de séquence, comme illustré dans la Figure 13, à la page 82. Toutefois, vous pouvez utiliser l'indicateur -e dans la commande `runmqsc` pour supprimer la sortie.
- Message d'erreur de syntaxe pour toutes les commandes détectées comme étant erronées.
- Un *message d'opérateur* indiquant le résultat de l'exécution de chaque commande. Par exemple, le message de l'opérateur indiquant que l'exécution d'une commande DEFINE QLOCAL a abouti est le suivant:

  ```
  AMQ8006: WebSphere MQ queue created.
  ```
- Autres messages résultant d'erreurs générales lors de l'exécution du fichier script.
- Bref récapitulatif statistique du rapport indiquant le nombre de commandes lues, le nombre de commandes comportant des erreurs de syntaxe et le nombre de commandes qui n'ont pas pu être traitées.

  Remarque : Le gestionnaire de files d'attente tente de traiter uniquement les commandes qui ne comportent aucune erreur de syntaxe.

```
Starting MQSC for queue manager jupiter.queue.manager.
 .
 .
     12:     DEFINE QLOCAL('ORANGE.LOCAL.QUEUE') REPLACE  +
       :            DESCR(' ') +
       :            PUT(ENABLED) +
       :            DEFPRTY(0) +
       :            DEFPSIST(NO) +
       :            GET(ENABLED) +
       :            MAXDEPTH(5000) +
       :            MAXMSGL(1024) +
       :            DEFSOPT(SHARED) +
       :            NOHARDENBO +
       :            USAGE(NORMAL) +
       :            NOTRIGGER;
AMQ8006: WebSphere MQ queue created.
       :
 .
 .
```

Figure 13. Extraction à partir d'un fichier de rapport de commandes MQSC

Exécution des fichiers de commandes MQSC fournis

Les fichiers de commandes MQSC suivants sont fournis avec WebSphere MQ:

amqscos0.tst
Définitions des objets utilisés par les exemples de programme.

amqscic0.tst
Définitions des files d'attente pour les transactions CICS .

Dans WebSphere MQ for Windows, ces fichiers se trouvent dans le répertoire *MQ_INSTALLATION_PATH*\tools\mqsc\samples. *MQ_INSTALLATION_PATH* représente le répertoire de haut niveau dans lequel WebSphere MQ est installé.

Sur les systèmes UNIX and Linux , ces fichiers se trouvent dans le répertoire *MQ_INSTALLATION_PATH*/samp. *MQ_INSTALLATION_PATH* représente le répertoire de haut niveau dans lequel WebSphere MQ est installé.

La commande qui les exécute est la suivante:

```
runmqsc < amqscos0.tst >test.out
```

Utilisation de runmqsc pour vérifier les commandes

Vous pouvez utiliser la commande runmqsc pour vérifier les commandes MQSC sur un gestionnaire de files d'attente local sans les exécuter réellement. Pour ce faire, définissez l'indicateur -v dans la commande runmqsc , par exemple:

```
runmqsc -v < myprog.in > myprog.out
```

Lorsque vous appelez runmqsc dans un fichier de commandes MQSC, le gestionnaire de files d'attente vérifie chaque commande et renvoie un rapport sans exécuter réellement les commandes MQSC. Cela vous permet de vérifier la syntaxe des commandes dans votre fichier de commandes. Ceci est particulièrement important si vous êtes:

- Exécution d'un grand nombre de commandes à partir d'un fichier de commandes.
- Utilisation d'un fichier de commandes MQSC plusieurs fois.

Le rapport renvoyé est similaire à celui présenté dans la Figure 13, à la page 82.

Vous ne pouvez pas utiliser cette méthode pour vérifier les commandes MQSC à distance. Par exemple, si vous tentez d'exécuter cette commande:

```
runmqsc -w 30 -v jupiter.queue.manager < myprog.in > myprog.out
```

l'indicateur -w, que vous utilisez pour indiquer que le gestionnaire de files d'attente est distant, est ignoré et la commande est exécutée localement en mode de vérification. 30 est le nombre de secondes pendant lesquelles WebSphere MQ attend les réponses du gestionnaire de files d'attente éloignées.

Exécution de commandes MQSC à partir de fichiers de traitement par lots

Si vous avez des commandes très longues ou que vous utilisez une séquence de commandes particulière à plusieurs reprises, envisagez de rediriger `stdin` à partir d'un fichier de traitement par lots.

Pour rediriger `stdin` à partir d'un fichier de traitement par lots, créez d'abord un fichier de traitement par lots contenant les commandes MQSC à l'aide de votre éditeur de texte habituel. Lorsque vous utilisez la commande `runmqsc` , utilisez les opérateurs de redirection. L'exemple suivant :

1. Crée un gestionnaire de files d'attente de test, TESTQM
2. Crée un CLNTCONN et un programme d'écoute correspondant pour utiliser le port TCP/IP 1600
3. Crée une file d'attente de test, TESTQ
4. Insère un message dans la file d'attente à l'aide de l'exemple de programme amqsputc

```
export MYTEMPQM=TESTQM
export MYPORT=1600
export MQCHLLIB=/var/mqm/qmgrs/$MQTEMPQM/@ipcc

crtmqm $MYTEMPQM
strmqm $MYTEMPQM
runmqlsr -m $MYTEMPQM -t TCP -p $MYPORT &

runmqsc $MYTEMPQM << EOF
  DEFINE CHANNEL(NTLM) CHLTYPE(SVRCONN) TRPTYPE(TCP)
  DEFINE CHANNEL(NTLM) CHLTYPE(CLNTCONN) QMNAME('$MYTEMPQM') CONNAME('hostname($MYPORT)')
  ALTER  CHANNEL(NTLM) CHLTYPE(CLNTCONN)
  DEFINE QLOCAL(TESTQ)
EOF

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

Figure 14. Exemple de script pour l'exécution de commandes MQSC à partir d'un fichier de traitement par lots

Résolution des problèmes liés aux commandes MQSC

Si vous ne parvenez pas à exécuter des commandes MQSC, utilisez les informations de cette rubrique pour voir si l'un de ces problèmes communs s'applique à vous. Il n'est pas toujours évident de savoir quel est le problème lorsque vous lisez l'erreur générée par une commande.

Si vous ne parvenez pas à exécuter les commandes MQSC, utilisez les informations suivantes pour voir si l'un de ces problèmes communs s'applique à vous. Il n'est pas toujours évident de savoir quel est le problème lorsque vous lisez l'erreur générée.

Lorsque vous utilisez la commande `runmqsc` , tenez compte des points suivants:

- Utilisez l'opérateur **<** pour rediriger l'entrée à partir d'un fichier. Si vous omettez cet opérateur, le gestionnaire de files d'attente interprète le nom de fichier comme un nom de gestionnaire de files d'attente et émet le message d'erreur suivant:

```
AMQ8118: WebSphere MQ queue manager does not exist.
```

- Si vous redirigez la sortie vers un fichier, utilisez l'opérateur de redirection **>** . Par défaut, le fichier est placé dans le répertoire de travail en cours lors de l'appel de `runmqsc` . Indiquez un nom de fichier qualifié complet pour envoyer votre sortie vers un fichier et un répertoire spécifiques.
- Vérifiez que vous avez créé le gestionnaire de files d'attente qui va exécuter les commandes, en utilisant la commande suivante pour afficher tous les gestionnaires de files d'attente:

```
dspmq
```

- Il doit être en cours d'exécution. Si tel n'est pas le cas, démarrez-le (voir Démarrage d'un gestionnaire de files d'attente). Un message d'erreur s'affiche si vous tentez de démarrer un gestionnaire de files d'attente déjà en cours d'exécution.
- Indiquez un nom de gestionnaire de files d'attente dans la commande `runmqsc` si vous n'avez pas défini de gestionnaire de files d'attente par défaut ou si vous obtenez l'erreur suivante:

```
AMQ8146: WebSphere MQ queue manager not available.
```

- Vous ne pouvez pas spécifier une commande MQSC en tant que paramètre de la commande `runmqsc` . Par exemple, ceci n'est pas valide:

```
runmqsc DEFINE QLOCAL(FRED)
```

- Vous ne pouvez pas entrer de commandes MQSC avant d'exécuter la commande `runmqsc` .
- Vous ne pouvez pas exécuter de commandes de contrôle à partir de `runmqsc`. Par exemple, vous ne pouvez pas exécuter la commande `strmqm` pour démarrer un gestionnaire de files d'attente alors que vous exécutez des commandes MQSC de manière interactive. Dans ce cas, vous recevez des messages d'erreur similaires aux suivants:

```
runmqsc
  .
  .
Starting MQSC for queue manager jupiter.queue.manager.

    1 : strmqm saturn.queue.manager
AMQ8405: Syntax error detected at or near end of cmd segment below:-s

AMQ8426: Valid MQSC commands are:
    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
     2 : end
```

Utilisation des gestionnaires de files d'attente

Exemples de commandes MQSC que vous pouvez utiliser pour afficher ou modifier des attributs de gestionnaire de files d'attente.

Affichage des attributs du gestionnaire de files d'attente

Pour afficher les attributs du gestionnaire de files d'attente spécifié dans la commande **runmqsc** , utilisez la commande MQSC suivante:

```
DISPLAY QMGR
```

La sortie standard de cette commande est illustrée dans la Figure 15, à la page 85

```
DISPLAY QMGR
     1 : DISPLAY QMGR
AMQ8408: Display Queue Manager details.
   QMNAME(QM1)                             ACCTCONO(DISABLED)
   ACCTINT(1800)                           ACCTMQI(OFF)
   ACCTQ(OFF)                              ACTIVREC(MSG)
   ACTVCONO (DISABLED)                     ACTVTRC (OFF)
   ALTDATE(2012-05-27)                     ALTTIME(16.14.01)
   AUTHOREV(DISABLED)                      CCSID(850)
   CHAD(DISABLED)                          CHADEV(DISABLED)
   CHADEXIT( )                             CHLEV(DISABLED)
   CLWLDATA( )                             CLWLEXIT( )
   CLWLLEN(100)                            CLWLMRUC(999999999)
   CLWLUSEQ(LOCAL)                         CMDEV(DISABLED)
   CMDLEVEL(750)                           COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
   CONFIGEV(DISABLED)                      CRDATE(2011-05-27)
   CRTIME(16.14.01)                        DEADQ()
   DEFXMITQ( )                             DESCR( )
   DISTL(YES)                              INHIBTEV(DISABLED)
   IPADDRV(IPV4)                           LOCALEV(DISABLED)
   LOGGEREV(DISABLED)                      MARKINT(5000)
   MAXHANDS(256)                           MAXMSGL(4194304)
   MAXPROPL(NOLIMIT)                       MAXPRTY(9)
   MAXUMSGS(10000)                         MONACLS(QMGR)
   MONCHL(OFF)                             MONQ(OFF)
   PARENT( )                               PERFMEV(DISABLED)
   PLATFORM(WINDOWSNT)                        PSRTYCNT(5)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSSYNCPT(IFPER)                         QMID(QM1_2011-05-27_16.14.01)
   PSMODE(ENABLED)                         REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   ROUTEREC(MSG)                           SCHINIT(QMGR)
   SCMDSERV(QMGR)                          SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             SSLKEYR(C:\Program Files\IBM\WebSphere
MQ\Data\qmgrs\QM1\ssl\key)
   SSLRKEYC(0)                             STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SYNCPT
   TREELIFE(1800)                          TRIGINT(999999999)
```

Figure 15. Sortie standard d'une commande DISPLAY QMGR

Le paramètre ALL est la valeur par défaut de la commande DISPLAY QMGR . Il affiche tous les attributs du gestionnaire de files d'attente. En particulier, la sortie vous indique le nom du gestionnaire de files d'attente par défaut, le nom de la file d'attente de rebut et le nom de la file d'attente de commandes.

Vous pouvez confirmer que ces files d'attente existent en entrant la commande suivante:

```
DISPLAY QUEUE (SYSTEM.*)
```

La liste des files d'attente correspondant au radical SYSTEM.*s'affiche. Les parenthèses sont obligatoires.

Modification des attributs du gestionnaire de files d'attente

Pour modifier les attributs du gestionnaire de files d'attente spécifié dans la commande **runmqsc** , utilisez la commande MQSC ALTER QMGRen spécifiant les attributs et les valeurs que vous souhaitez modifier. Par exemple, utilisez les commandes suivantes pour modifier les attributs de `jupiter.queue.manager`:

```
runmqsc jupiter.queue.manager
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

La commande ALTER QMGR modifie la file d'attente de rebut utilisée et active les événements d'interdiction.

Référence associée

Attributs du gestionnaire de files d'attente

Utilisation des files d'attente locales

Cette section contient des exemples de certaines commandes MQSC que vous pouvez utiliser pour gérer les files d'attente locales, de modèle et d'alias.

Pour plus d'informations sur ces commandes, voir Référence MQSC .

Définition d'une file d'attente locale

Pour une application, le gestionnaire de files d'attente local est le gestionnaire de files d'attente auquel l'application est connectée. Les files d'attente gérées par le gestionnaire de files d'attente local sont dites locales pour ce gestionnaire de files d'attente.

Utilisez la commande MQSC DEFINE QLOCAL pour créer une file d'attente locale. Vous pouvez également utiliser la valeur par défaut définie dans la définition de file d'attente locale par défaut ou modifier les caractéristiques de la file d'attente locale par défaut.

Remarque : La file d'attente locale par défaut est nommée SYSTEM.DEFAULT.LOCAL.QUEUE et a été créée sur l'installation du système.

Par exemple, la commande DEFINE QLOCAL qui suit définit une file d'attente appelée ORANGE.LOCAL.QUEUE avec les caractéristiques suivantes:

- Il est activé pour les requêtes, activé pour les insertions et fonctionne en fonction de l'ordre de priorité.
- Il s'agit d'une file d'attente *normale* ; il ne s'agit pas d'une file d'attente d'initialisation ou de transmission et elle ne génère pas de messages de déclenchement.
- La longueur maximale de la file d'attente est de 5000 messages ; la longueur maximale est de 4194304 octets.

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
       DESCR('Queue for messages from other systems') +
       PUT (ENABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (PRIORITY) +
       MAXDEPTH (5000) +
       MAXMSGL (4194304) +
       USAGE (NORMAL);
```

Remarque :

1. A l'exception de la valeur de la description, toutes les valeurs d'attribut affichées sont les valeurs par défaut. Nous les avons montrées ici à des fins d'illustration. Vous pouvez les omettre si vous êtes sûr que les valeurs par défaut correspondent à celles que vous souhaitez ou qu'elles n'ont pas été modifiées. Voir aussi «Affichage des attributs d'objet par défaut», à la page 87.
2. USAGE (NORMAL) indique que cette file d'attente n'est pas une file d'attente de transmission.

3. Si vous disposez déjà d'une file d'attente locale sur le même gestionnaire de files d'attente avec le nom ORANGE.LOCAL.QUEUE, cette commande échoue. Utilisez l'attribut REPLACE si vous souhaitez remplacer la définition existante d'une file d'attente, mais voir aussi «Modification des attributs de file d'attente locale», à la page 88.

Définition d'une file d'attente de rebut

Chaque gestionnaire de files d'attente doit disposer d'une file d'attente locale à utiliser comme file d'attente de rebut pour que les messages qui ne peuvent pas être distribués à leur destination correcte puissent être stockés en vue d'une extraction ultérieure. Vous devez indiquer au gestionnaire de files d'attente la file d'attente de rebut.

Pour informer le gestionnaire de files d'attente de la file d'attente de rebut, indiquez un nom de file d'attente de rebut dans la commande `crtmqm` (`crtmqm -u DEAD.LETTER.QUEUE`, par exemple) ou en utilisant l'attribut DEADQ de la commande ALTER QMGR pour en spécifier une ultérieurement. Vous devez définir la file d'attente de rebut avant de l'utiliser.

Exemple de file d'attente de rebut appelée SYSTEM.DEAD.LETTER.QUEUE est disponible avec le produit. Cette file d'attente est automatiquement créée lorsque vous créez le gestionnaire de files d'attente. Vous pouvez modifier cette définition si nécessaire et la renommer.

Une file d'attente de rebut n'a pas d'exigences particulières si ce n'est que:

- Il doit s'agir d'une file d'attente locale
- Son attribut MAXMSGL (longueur maximale de message) doit permettre à la file d'attente de recevoir les messages les plus volumineux que le gestionnaire de files d'attente doit traiter **plus** la taille de l'en-tête de message non livrés (MQDLH)

WebSphere MQ fournit un gestionnaire de files d'attente de rebut qui permet d'indiquer comment les messages trouvés dans une file d'attente de rebut doivent être traités ou supprimés. Pour plus d'informations, voir Gestion des messages non livrés avec le gestionnaire de files d'attente de rebut WebSphere MQ .

Affichage des attributs d'objet par défaut

Vous pouvez utiliser la commande DISPLAY QUEUE pour afficher les attributs qui ont été extraits de l'objet par défaut lorsqu'un objet WebSphere MQ a été défini.

Lorsque vous définissez un objet WebSphere MQ , il prend tous les attributs que vous ne spécifiez pas dans l'objet par défaut. Par exemple, lorsque vous définissez une file d'attente locale, celle-ci hérite de tous les attributs que vous omettez dans la définition de la file d'attente locale par défaut, appelée SYSTEM.DEFAULT.LOCAL.QUEUE. Pour voir exactement quels sont ces attributs, utilisez la commande suivante:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

La syntaxe de cette commande est différente de celle de la commande DEFINE correspondante. Dans la commande DISPLAY, vous pouvez indiquer uniquement le nom de la file d'attente, tandis que dans la commande DEFINE, vous devez indiquer le type de la file d'attente, à savoir QLOCAL, QALIAS, QMODEL ou QREMOTE.

Vous pouvez afficher les attributs de manière sélective en les spécifiant individuellement. Exemple :

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
        MAXDEPTH +
        MAXMSGL +
        CURDEPTH;
```

Cette commande affiche les trois attributs spécifiés comme suit:

```
AMQ8409: Display Queue details.
   QUEUE(ORANGE.LOCAL.QUEUE)               TYPE(QLOCAL)
   CURDEPTH(0)                             MAXDEPTH(5000)
```

```
   MAXMSGL(4194304)
```

CURDEPTH est la longueur de la file d'attente en cours, c'est-à-dire le nombre de messages dans la file d'attente. Il s'agit d'un attribut utile à afficher, car en surveillant la longueur de la file d'attente, vous pouvez vous assurer que la file d'attente n'est pas saturée.

Copie d'une définition de file d'attente locale

Vous pouvez copier une définition de file d'attente à l'aide de l'attribut LIKE de la commande DEFINE.

Exemple :

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

Cette commande crée une file d'attente avec les mêmes attributs que notre file d'attente d'origine ORANGE.LOCAL.QUEUE, plutôt que celles de la file d'attente locale par défaut du système. Entrez le nom de la file d'attente à copier ***exactement*** tel qu'il a été entré lors de la création de la file d'attente. Si le nom contient des caractères en minuscules, placez-le entre apostrophes.

Vous pouvez également utiliser cette forme de la commande DEFINE pour copier une définition de file d'attente, mais remplacer une ou plusieurs modifications apportées aux attributs de l'original. Exemple :

```
DEFINE QLOCAL (THIRD.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE) +
       MAXMSGL(1024);
```

Cette commande copie les attributs de la file d'attente ORANGE.LOCAL.QUEUE dans la file d'attente THIRD.QUEUE, mais spécifie que la longueur maximale des messages dans la nouvelle file d'attente doit être de 1024 octets au lieu de 4194304.

Remarque :

1. Lorsque vous utilisez l'attribut LIKE dans une commande DEFINE, vous copiez uniquement les attributs de file d'attente. Vous ne copiez pas les messages de la file d'attente.
2. Si vous définissez une file d'attente locale, sans spécifier LIKE, elle est identique à DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE).

Modification des attributs de file d'attente locale

Vous pouvez modifier les attributs de file d'attente de deux manières, à l'aide de la commande ALTER QLOCAL ou de la commande DEFINE QLOCAL avec l'attribut REPLACE.

Dans «Définition d'une file d'attente locale», à la page 86, la file d'attente appelée ORANGE.LOCAL.QUEUE a été définie. Supposons, par exemple, que vous souhaitiez réduire la longueur maximale des messages de cette file d'attente à 10 000 octets.

- A l'aide de la commande ALTER:

  ```
  ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
  ```

  Cette commande modifie un attribut unique, celui de la longueur maximale de message ; tous les autres attributs restent identiques.
- Utilisation de la commande DEFINE avec l'option REPLACE, par exemple:

  ```
  DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
  ```

Cette commande modifie non seulement la longueur maximale des messages, mais également tous les autres attributs auxquels sont affectées leurs valeurs par défaut. La file d'attente est maintenant mise en file d'attente activée alors qu'auparavant elle était mise en file d'attente interdite. L'insertion activée est la valeur par défaut, comme indiqué par la file d'attente SYSTEM.DEFAULT.LOCAL.QUEUE.

Si vous ***réduisez*** la longueur maximale des messages dans une file d'attente existante, les messages existants ne sont pas affectés. Toutefois, tout nouveau message doit répondre aux nouveaux critères.

Effacement d'une file d'attente locale

Vous pouvez utiliser la commande CLEAR pour effacer une file d'attente locale.

Pour supprimer tous les messages d'une file d'attente locale appelée MAGENTA.QUEUE, utilisez la commande suivante:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

Remarque : Aucune invite ne vous permet de changer d'avis. Une fois que vous appuyez sur la touche Entrée, les messages sont perdus.

Vous ne pouvez pas effacer une file d'attente si:

- Des messages non validés ont été placés dans la file d'attente sous le point de synchronisation.
- La file d'attente est ouverte par une application.

Suppression d'une file d'attente locale

Vous pouvez utiliser la commande MQSC DELETE QLOCAL pour supprimer une file d'attente locale.

Une file d'attente ne peut pas être supprimée si elle contient des messages non validés. Toutefois, si la file d'attente contient un ou plusieurs messages validés et aucun message non validé, elle ne peut être supprimée que si vous spécifiez l'option PURGE. Exemple :

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

La spécification de NOPURGE à la place de PURGE garantit que la file d'attente n'est pas supprimée si elle contient des messages validés.

Exploration des files d'attente

WebSphere MQ fournit un exemple de navigateur de files d'attente que vous pouvez utiliser pour examiner le contenu des messages d'une file d'attente. Le navigateur est fourni dans les formats source et exécutable.

MQ_INSTALLATION_PATH représente le répertoire de haut niveau dans lequel WebSphere MQ est installé.

Dans WebSphere MQ for Windows, les noms de fichier et les chemins de l'exemple de navigateur de files d'attente sont les suivants:

Source
MQ_INSTALLATION_PATH\tools\c\samples\

Exécutable
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

Dans WebSphere MQ for UNIX and Linux, les noms de fichier et les chemins d'accès sont les suivants:

Source
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

Exécutable
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

L'exemple requiert deux paramètres d'entrée, le nom de la file d'attente et le nom du gestionnaire de files d'attente. Exemple :

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

Les résultats typiques de cette commande sont présentés dans Figure 16, à la page 90.

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 850
  Format : 'MQEVENT '
  Priority : 0  Persistence : 0
  MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                    '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '104'

****   Message      ****

 length - 104 bytes

00000000:  0700 0000 2400 0000 0100 0000 2C00 0000 '....→.......,...'
00000010:  0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020:  0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030:  0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040:  7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager    '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020                     '                '

 No more messages
 MQCLOSE
 MQDISC
```

Figure 16. Résultats typiques du navigateur de files d'attente

Activation des files d'attente volumineuses

IBM WebSphere MQ prend en charge les files d'attente supérieures à 2 Go.

Sur les systèmes Windows , la prise en charge des fichiers volumineux est disponible sans activation supplémentaire. Sur les systèmes AIX, HP-UX, Linuxet Solaris , vous devez activer explicitement la prise en charge des fichiers volumineux avant de pouvoir créer des fichiers de file d'attente de plus de 2 Go.

Consultez la documentation de votre système d'exploitation pour plus d'informations sur la procédure à suivre.

Certains utilitaires, tels que tar, ne peuvent pas gérer les fichiers de plus de 2 Go. Avant d'activer la prise en charge des fichiers volumineux, consultez la documentation de votre système d'exploitation pour plus d'informations sur les restrictions relatives aux utilitaires que vous utilisez.

Pour plus d'informations sur la planification de la quantité de stockage dont vous avez besoin pour les files d'attente, visitez le site Web IBM WebSphere MQ pour obtenir des rapports de performances spécifiques à la plateforme:

```
https://www.ibm.com/software/integration/ts/mqseries/
```

Utilisation de files d'attente alias

Vous pouvez définir une file d'attente alias pour faire référence indirectement à une autre file d'attente ou rubrique.

V 7.5.0.8

Avertissement : Les listes de distribution ne prennent pas en charge les files d'attente alias qui pointent vers des objets de rubrique. Depuis la Version 7.5.0, Fix Pack 8, si une file d'attente alias pointe vers un objet de rubrique dans une liste de distribution, IBM WebSphere MQ renvoie MQRC_ALIAS_BASE_Q_TYPE_ERROR.

La file d'attente à laquelle une file d'attente alias fait référence peut être l'une des suivantes:

- Une file d'attente locale (voir «Définition d'une file d'attente locale», à la page 86).
- Définition locale d'une file d'attente éloignée (voir «Création d'une définition locale d'une file d'attente éloignée», à la page 116).
- Rubrique.

Une file d'attente alias n'est pas une file d'attente réelle, mais une définition qui se résout en file d'attente réelle (ou cible) lors de l'exécution. La définition de file d'attente alias indique la file d'attente cible. Lorsqu'une application effectue un appel MQOPEN à une file d'attente alias, le gestionnaire de files d'attente résout l'alias en nom de file d'attente cible.

Une file d'attente alias ne peut pas être résolue en une autre file d'attente alias définie localement. Toutefois, une file d'attente alias peut être convertie en files d'attente alias définies ailleurs dans des clusters dont le gestionnaire de files d'attente local est membre. Pour plus d'informations, voir Résolution de nom .

Les files d'attente alias sont utiles pour:

- Attribution à différentes applications de différents niveaux de droits d'accès à la file d'attente cible.
- Permettre à différentes applications de travailler avec la même file d'attente de différentes manières. (Vous souhaitez peut-être affecter des priorités par défaut différentes ou des valeurs de persistance par défaut différentes.)
- Simplification de la maintenance, de la migration et de l'équilibrage de la charge de travail. (Vous souhaitez peut-être modifier le nom de la file d'attente cible sans avoir à modifier votre application, qui continue d'utiliser l'alias.)

Par exemple, supposons qu'une application ait été développée pour placer des messages dans une file d'attente appelée MY.ALIAS.QUEUE. Il indique le nom de cette file d'attente lorsqu'il émet une demande MQOPEN et, indirectement, s'il insère un message dans cette file d'attente. L'application ne sait pas que la file d'attente est une file d'attente d'alias. Pour chaque appel MQI utilisant cet alias, le gestionnaire de files d'attente résout le nom de la file d'attente réelle, qui peut être une file d'attente locale ou une file d'attente éloignée définie sur ce gestionnaire de files d'attente.

En modifiant la valeur de l'attribut TARGET, vous pouvez rediriger les appels MQI vers une autre file d'attente, éventuellement sur un autre gestionnaire de files d'attente. Cela est utile pour la maintenance, la migration et l'équilibrage de charge.

Définition d'une file d'attente alias

La commande suivante crée une file d'attente alias:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

Cette commande redirige les appels MQI qui spécifient MY.ALIAS.QUEUE dans la file d'attente YELLOW.QUEUE. La commande ne crée pas la file d'attente cible ; les appels MQI échouent si la file d'attente est YELLOW.QUEUE n'existe pas lors de l'exécution.

Si vous modifiez la définition d'alias, vous pouvez rediriger les appels MQI vers une autre file d'attente. Exemple :

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

Cette commande redirige les appels MQI vers une autre file d'attente, MAGENTA.QUEUE.

Vous pouvez également utiliser des files d'attente alias pour qu'une seule file d'attente (la file d'attente cible) ait des attributs différents pour différentes applications. Pour ce faire, vous définissez deux alias, un pour chaque application. Supposons qu'il existe deux applications:

- L'application ALPHA peut placer des messages sur YELLOW.QUEUE, mais n'est pas autorisé à en extraire des messages.
- L'application bêta peut recevoir des messages de YELLOW.QUEUE, mais n'est pas autorisé à y placer des messages.

La commande suivante définit un alias qui est activé et désactivé pour l'application ALPHA:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (ENABLED) +
       GET (DISABLED)
```

La commande suivante définit un alias qui est mis à l'état désactivé et activé pour l'application BETA:

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (DISABLED) +
       GET (ENABLED)
```

ALPHA utilise le nom de file d'attente ALPHAS.ALIAS.QUEUE dans ses appels MQI ; BETA utilise le nom de file d'attente BETAS.ALIAS.QUEUE. Ils accèdent tous deux à la même file d'attente, mais de différentes manières.

Vous pouvez utiliser les attributs LIKE et REPLACE lorsque vous définissez des alias de file d'attente, de la même manière que vous utilisez ces attributs avec des files d'attente locales.

Utilisation d'autres commandes avec des files d'attente alias

Vous pouvez utiliser les commandes MQSC appropriées pour afficher ou modifier les attributs de file d'attente alias ou pour supprimer l'objet file d'attente alias. Exemple :

Utilisez la commande suivante pour afficher les attributs de la file d'attente alias:

```
DISPLAY QALIAS (ALPHAS.ALIAS.QUEUE)
```

Utilisez la commande suivante pour modifier le nom de la file d'attente de base, dans laquelle l'alias est résolu, où l'option `force` force la modification même si la file d'attente est ouverte:

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGET(ORANGE.LOCAL.QUEUE) FORCE
```

Utilisez la commande suivante pour supprimer cet alias de file d'attente:

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

Vous ne pouvez pas supprimer une file d'attente alias si la file d'attente est ouverte pour une application. Pour plus d'informations sur cette commande et sur d'autres commandes de file d'attente alias, voir Référence MQSC .

Utilisation des files d'attente modèles

Un gestionnaire de files d'attente crée une *file d'attente dynamique* s'il reçoit un appel MQI d'une application spécifiant un nom de file d'attente qui a été défini en tant que file d'attente modèle. Le nom de la nouvelle file d'attente dynamique est généré par le gestionnaire de files d'attente lors de la création de la file d'attente. Une *file d'attente modèle* est un modèle qui spécifie les attributs des files d'attente dynamiques créées à partir de celle-ci. Les files d'attente modèles fournissent une méthode pratique pour les applications afin de créer des files d'attente selon les besoins.

Définition d'une file d'attente modèle

Vous définissez une file d'attente modèle avec un ensemble d'attributs de la même manière que vous définissez une file d'attente locale. Les files d'attente modèles et les files d'attente locales ont le même ensemble d'attributs, sauf que dans les files d'attente modèles, vous pouvez indiquer si les files d'attente dynamiques créées sont temporaires ou permanentes. (Les files d'attente permanentes sont gérées lors des redémarrages du gestionnaire de files d'attente, alors que les files d'attente temporaires ne le sont pas.) Exemple :

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
       DESCR('Queue for messages from application X') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE (NORMAL) +
       DEFTYPE (PERMDYN)
```

Cette commande crée une définition de file d'attente modèle. A partir de l'attribut DEFTYPE, vous pouvez voir que les files d'attente réelles créées à partir de ce modèle sont des files d'attente dynamiques permanentes. Tous les attributs non spécifiés sont automatiquement copiés à partir de SYSYTEM.DEFAULT.MODEL.QUEUE .

Vous pouvez utiliser les attributs LIKE et REPLACE lorsque vous définissez des files d'attente modèles, de la même manière que vous les utilisez avec des files d'attente locales.

Utilisation d'autres commandes avec des files d'attente modèles

Vous pouvez utiliser les commandes MQSC appropriées pour afficher ou modifier les attributs d'une file d'attente modèle ou pour supprimer l'objet de file d'attente modèle. Exemple :

Utilisez la commande suivante pour afficher les attributs de la file d'attente modèle:

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

Utilisez la commande suivante pour modifier le modèle afin d'activer les insertions dans les files d'attente dynamiques créées à partir de ce modèle:

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

Utilisez la commande suivante pour supprimer cette file d'attente modèle:

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

Utilisation des rubriques d'administration

Utilisez les commandes MQSC pour gérer les rubriques d'administration.

Pour plus d'informations sur ces commandes, voir Référence MQSC .

Concepts associés

Objets de rubrique d'administration

«Définition d'une rubrique d'administration», à la page 94
Utilisez la commande MQSC **DEFINE TOPIC** pour créer une rubrique d'administration. Lorsque vous définissez une rubrique d'administration, vous pouvez éventuellement définir chaque attribut de rubrique.

«Affichage des attributs d'objet de rubrique d'administration», à la page 95
La commande MQSC **DISPLAY TOPIC** permet d'afficher un objet de rubrique d'administration.

«Modification des attributs de rubrique d'administration», à la page 95
Vous pouvez modifier les attributs de rubrique de deux manières, à l'aide de la commande **ALTER TOPIC** ou de la commande **DEFINE TOPIC** avec l'attribut **REPLACE** .

«Copie d'une définition de rubrique d'administration», à la page 96
Vous pouvez copier une définition de rubrique à l'aide de l'attribut LIKE de la commande **DEFINE** .

«Suppression d'une définition de rubrique d'administration», à la page 96
Vous pouvez utiliser la commande MQSC **DELETE TOPIC** pour supprimer une rubrique d'administration.

Définition d'une rubrique d'administration

Utilisez la commande MQSC **DEFINE TOPIC** pour créer une rubrique d'administration. Lorsque vous définissez une rubrique d'administration, vous pouvez éventuellement définir chaque attribut de rubrique.

Tout attribut de la rubrique qui n'est pas explicitement défini est hérité de la rubrique d'administration par défaut, SYSTEM.DEFAULT.TOPIC, qui a été créé lors de l'installation du système.

Par exemple, la commande **DEFINE TOPIC** qui suit, définit une rubrique appelée **ORANGE.TOPIC** avec les caractéristiques suivantes:

- Se résout en la chaîne de rubrique ORANGE. Pour plus d'informations sur la façon dont les chaînes de rubrique peuvent être utilisées, voir Combinaison de chaînes de rubrique.
- Tout attribut défini sur ASPARENT utilise l'attribut tel que défini par la rubrique parent de cette rubrique. Cette action est répétée dans l'arborescence de rubriques jusqu'à la rubrique racine, SYSTEM.BASE.TOPIC a été trouvé. Pour plus d'informations sur les arborescences de rubriques, voir Arborescences de rubriques.

```
DEFINE TOPIC (ORANGE.TOPIC) +
       TOPICSTR (ORANGE) +
       DEFPRTY(ASPARENT) +
       NPMSGDLV(ASPARENT)
```

Remarque :

- A l'exception de la valeur de la chaîne de rubrique, toutes les valeurs d'attribut affichées sont les valeurs par défaut. Ils ne sont présentés ici qu'à titre d'illustration. Vous pouvez les omettre si vous

êtes sûr que les valeurs par défaut correspondent à celles que vous souhaitez ou qu'elles n'ont pas été modifiées. Voir aussi «Affichage des attributs d'objet de rubrique d'administration», à la page 95.

- Si vous disposez déjà d'une rubrique d'administration sur le même gestionnaire de files d'attente avec le nom ORANGE.TOPIC, cette commande échoue. Utilisez l'attribut REPLACE si vous souhaitez remplacer la définition existante d'une rubrique, mais voir aussi «Modification des attributs de rubrique d'administration», à la page 95

Affichage des attributs d'objet de rubrique d'administration

La commande MQSC **DISPLAY TOPIC** permet d'afficher un objet de rubrique d'administration.

Pour afficher toutes les rubriques, utilisez:

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

Vous pouvez afficher les attributs de manière sélective en les spécifiant individuellement.Par exemple :

```
DISPLAY TOPIC(ORANGE.TOPIC) +
        TOPICSTR +
        DEFPRTY +
        NPMSGDLV
```

Cette commande affiche les trois attributs spécifiés comme suit:

```
AMQ8633: Display topic details.
   TOPIC(ORANGE.TOPIC)                     TYPE(LOCAL)
   TOPICSTR(ORANGE)                        DEFPRTY(ASPARENT)
   NPMSGDLV(ASPARENT)
```

Pour afficher les valeurs de la rubrique ASPARENT telles qu'elles sont utilisées lors de l'exécution, utilisez DISPLAY TPSTATUS. Par exemple, utilisez :

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

La commande affiche les détails suivants:

```
AMQ8754: Display topic status details.
   TOPICSTR(ORANGE)                        DEFPRTY(0)
   NPMSGDLV(ALLAVAIL)
```

Lorsque vous définissez une rubrique d'administration, elle prend tous les attributs que vous ne spécifiez pas explicitement à partir de la rubrique d'administration par défaut, appelée SYSTEM.DEFAULT.TOPIC. Pour voir quels sont ces attributs par défaut, utilisez la commande suivante:

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

Modification des attributs de rubrique d'administration

Vous pouvez modifier les attributs de rubrique de deux manières, à l'aide de la commande **ALTER TOPIC** ou de la commande **DEFINE TOPIC** avec l'attribut **REPLACE** .

Par exemple, si vous souhaitez modifier la priorité par défaut des messages distribués à une rubrique appelée ORANGE.TOPIC, à la valeur 5, utilisez l'une des commandes suivantes.

- A l'aide de la commande **ALTER** :

  ```
  ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
  ```

  Cette commande modifie à 5 un attribut unique, celui de la priorité par défaut des messages distribués à cette rubrique ; tous les autres attributs restent identiques.

- A l'aide de la commande **DEFINE** :

  ```
  DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
  ```

Cette commande modifie la priorité par défaut des messages distribués à cette rubrique. Tous les autres attributs reçoivent leurs valeurs par défaut.

Si vous modifiez la priorité des messages envoyés à cette rubrique, les messages existants ne sont pas affectés. Tout nouveau message, cependant, utilise la priorité spécifiée si elle n'est pas fournie par l'application de publication.

Copie d'une définition de rubrique d'administration

Vous pouvez copier une définition de rubrique à l'aide de l'attribut LIKE de la commande **DEFINE** .

Exemple :

```
DEFINE TOPIC (MAGENTA.TOPIC) +
       LIKE (ORANGE.TOPIC)
```

Cette commande crée une rubrique, MAGENTA.TOPIC, avec les mêmes attributs que la rubrique d'origine, ORANGE.TOPIC, plutôt que ceux de la rubrique d'administration par défaut du système. Entrez le nom de la rubrique à copier exactement tel qu'il a été entré lors de la création de la rubrique. Si le nom contient des caractères minuscules, placez-le entre apostrophes.

Vous pouvez également utiliser cette forme de la commande **DEFINE** pour copier une définition de rubrique, mais apporter des modifications aux attributs de l'original. Exemple :

```
DEFINE TOPIC(BLUE.TOPIC) +
       TOPICSTR(BLUE) +
       LIKE(ORANGE.TOPIC)
```

Vous pouvez également copier les attributs de la rubrique BLUE.TOPIC à la rubrique GREEN.TOPIC et indiquez que lorsque les publications ne peuvent pas être distribuées à leur file d'attente de souscription correcte, elles ne sont pas placées dans la file d'attente de rebut. Exemple :

```
DEFINE TOPIC(GREEN.TOPIC) +
       TOPICSTR(GREEN) +
       LIKE(BLUE.TOPIC) +
       USEDLQ(NO)
```

Suppression d'une définition de rubrique d'administration

Vous pouvez utiliser la commande MQSC **DELETE TOPIC** pour supprimer une rubrique d'administration.

```
DELETE TOPIC(ORANGE.TOPIC)
```

Les applications ne pourront plus ouvrir la rubrique pour la publication ou créer de nouveaux abonnements à l'aide du nom d'objet ORANGE.TOPIC. Les applications de publication pour lesquelles la rubrique est ouverte peuvent continuer à publier la chaîne de rubrique résolue. Tous les abonnements déjà effectués à cette rubrique continuent de recevoir des publications après la suppression de la rubrique.

Les applications qui ne font pas référence à cet objet de rubrique mais qui utilisent la chaîne de rubrique résolue représentée par cet objet de rubrique,'ORANGE'dans cet exemple, continuent de fonctionner. Dans ce cas, ils héritent des propriétés d'un objet de rubrique plus haut dans l'arborescence de rubriques. Pour plus d'informations sur les arborescences de rubriques, voir Arborescences de rubriques.

Utilisation des abonnements

Utilisez les commandes MQSC pour gérer les abonnements.

Les abonnements peuvent être de l'un des trois types définis dans l'attribut **SUBTYPE** :

ADMIN
Défini administrativement par un utilisateur.

Proxy
Abonnement créé en interne pour le routage des publications entre les gestionnaires de files d'attente.

API
Créé à l'aide d'un programme, par exemple, à l'aide de l'appel MQI MQSUB.

Pour plus d'informations sur ces commandes, voir Référence MQSC .

Concepts associés

«Définition d'un abonnement d'administration», à la page 97
Utilisez la commande MQSC **DEFINE SUB** pour créer un abonnement d'administration. Vous pouvez également utiliser la valeur par défaut définie dans la définition d'abonnement local par défaut. Vous pouvez également modifier les caractéristiques de l'abonnement à partir de celles de l'abonnement local par défaut, SYSTEM.DEFAULT.SUB créée lors de l'installation du système.

«Affichage des attributs des abonnements», à la page 98
Vous pouvez utiliser la commande **DISPLAY SUB** pour afficher les attributs configurés de tout abonnement connu du gestionnaire de files d'attente.

«Modification des attributs d'abonnement local», à la page 99
Vous pouvez modifier les attributs d'abonnement de deux manières, à l'aide de la commande **ALTER SUB** ou de la commande **DEFINE SUB** avec l'attribut **REPLACE** .

«Copie d'une définition d'abonnement local», à la page 99
Vous pouvez copier une définition d'abonnement à l'aide de l'attribut **LIKE** de la commande **DEFINE** .

«Suppression d'un abonnement», à la page 99
Vous pouvez utiliser la commande MQSC **DELETE SUB** pour supprimer un abonnement local.

Définition d'un abonnement d'administration

Utilisez la commande MQSC **DEFINE SUB** pour créer un abonnement d'administration. Vous pouvez également utiliser la valeur par défaut définie dans la définition d'abonnement local par défaut. Vous pouvez également modifier les caractéristiques de l'abonnement à partir de celles de l'abonnement local par défaut, SYSTEM.DEFAULT.SUB créée lors de l'installation du système.

Par exemple, la commande **DEFINE SUB** qui suit définit un abonnement appelé ORANGE avec les caractéristiques suivantes:

- Abonnement durable, c'est-à-dire qu'il est conservé après le redémarrage du gestionnaire de files d'attente, avec un délai d'expiration illimité.
- Réception des publications effectuées sur la chaîne de rubrique ORANGE, avec les priorités de message définies par les applications de publication.
- Les publications fournies pour cet abonnement sont envoyées à la file d'attente locale SUBQ. Cette file d'attente doit être définie avant la définition de l'abonnement.

```
DEFINE SUB (ORANGE) +
       TOPICSTR (ORANGE) +
       DESTCLAS (PROVIDED) +
       DEST (SUBQ) +
       EXPIRY (UNLIMITED) +
       PUBPRTY (ASPUB)
```

Remarque :

- L'abonnement et le nom de la chaîne de rubrique ne doivent pas nécessairement correspondre.
- A l'exception des valeurs de la description et de la chaîne de rubrique, toutes les valeurs d'attribut affichées sont les valeurs par défaut. Ils ne sont présentés ici qu'à titre d'illustration. Vous pouvez les omettre si vous êtes sûr que les valeurs par défaut correspondent à celles que vous souhaitez ou qu'elles n'ont pas été modifiées. Voir aussi «Affichage des attributs des abonnements», à la page 98.
- Si vous disposez déjà d'un abonnement local sur le même gestionnaire de files d'attente avec le nom TEST, cette commande échoue. Utilisez l'attribut **REPLACE** si vous souhaitez remplacer la définition

existante d'une file d'attente, mais voir aussi «Modification des attributs d'abonnement local», à la page 99.

- Si la file d'attente SUBQ n'existe pas, cette commande échoue.

Affichage des attributs des abonnements

Vous pouvez utiliser la commande **DISPLAY SUB** pour afficher les attributs configurés de tout abonnement connu du gestionnaire de files d'attente.

Par exemple, utilisez :

```
DISPLAY SUB (ORANGE)
```

Vous pouvez afficher les attributs de manière sélective en les spécifiant individuellement. Exemple :

```
DISPLAY SUB (ORANGE) +
        SUBID +
        TOPICSTR +
        DURABLE
```

Cette commande affiche les trois attributs spécifiés comme suit:

```
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

TOPICSTR est la chaîne de rubrique résolue sur laquelle cet abonné fonctionne. Lorsqu'un abonnement est défini pour utiliser un objet de rubrique, la chaîne de rubrique de cet objet est utilisée comme préfixe de la chaîne de rubrique fournie lors de la création de l'abonnement. SUBID est un identificateur unique attribué par le gestionnaire de files d'attente lors de la création d'un abonnement. Il s'agit d'un attribut utile à afficher car certains noms d'abonnement peuvent être longs ou dans des jeux de caractères différents pour lesquels il peut devenir impossible de les afficher.

Une autre méthode d'affichage des abonnements consiste à utiliser le SUBID:

```
DISPLAY SUB +
        SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
        TOPICSTR +
        DURABLE
```

Cette commande génère la même sortie que précédemment:

```
AMQ8096: WebSphere MQ subscription inquired.
 SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

Les abonnements de proxy sur un gestionnaire de files d'attente ne sont pas affichés par défaut. Pour les afficher, spécifiez un **SUBTYPE** de type PROXY ou ALL.

Vous pouvez utiliser la commande DISPLAY SBSTATUS pour afficher les attributs d'exécution. Par exemple, utilisez la commande suivante:

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

La sortie suivante s'affiche :

```
AMQ8099: WebSphere MQ subscription status inquired.
   SUB(ORANGE)
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   NUMMSGS(0)
```

Lorsque vous définissez un abonnement d'administration, il prend tous les attributs que vous ne spécifiez pas explicitement à partir de l'abonnement par défaut, appelé SYSTEM.DEFAULT.SUB. Pour voir quels sont ces attributs par défaut, utilisez la commande suivante:

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

Modification des attributs d'abonnement local

Vous pouvez modifier les attributs d'abonnement de deux manières, à l'aide de la commande **ALTER SUB** ou de la commande **DEFINE SUB** avec l'attribut **REPLACE** .

Si, par exemple, vous souhaitez modifier la priorité des messages distribués à un abonnement appelé ORANGE sur 5, utilisez l'une des commandes suivantes:

- A l'aide de la commande ALTER:

  ```
  ALTER SUB(ORANGE) PUBPRTY(5)
  ```

  Cette commande modifie un attribut unique, celui de la priorité des messages distribués à cet abonnement à 5 ; tous les autres attributs restent les mêmes.
- A l'aide de la commande DEFINE:

  ```
  DEFINE SUB (ORANGE) PUBPRTY(5) REPLACE
  ```

  Cette commande modifie non seulement la priorité des messages distribués à cet abonnement, mais également tous les autres attributs auxquels sont attribuées leurs valeurs par défaut.

Si vous modifiez la priorité des messages envoyés à cet abonnement, les messages existants ne sont pas affectés. Toutefois, tous les nouveaux messages ont la priorité spécifiée.

Copie d'une définition d'abonnement local

Vous pouvez copier une définition d'abonnement à l'aide de l'attribut **LIKE** de la commande **DEFINE** .

Exemple :

```
DEFINE SUB (BLUE) +
       LIKE (ORANGE)
```

Vous pouvez également copier les attributs de la sous-commande REAL dans la sous-commande THIRD.SUBet indiquer que l' correlID des publications distribuées est THIRD, plutôt que l' correlIDdes diffuseurs de publications. Exemple :

```
DEFINE SUB(THIRD.SUB) +
       LIKE(BLUE) +
       DESTCORL(ORANGE)
```

Suppression d'un abonnement

Vous pouvez utiliser la commande MQSC **DELETE SUB** pour supprimer un abonnement local.

```
DELETE SUB(ORANGE)
```

Vous pouvez également supprimer un abonnement à l'aide du SUBID:

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

Vérification des messages sur un abonnement

Pourquoi et quand exécuter cette tâche

Lorsqu'un abonnement est défini, il est associé à une file d'attente. Les messages publiés correspondant à cet abonnement sont placés dans cette file d'attente.

Notez que les commandes **runmqsc** suivantes affichent uniquement les abonnements qui ont reçu des messages.

Pour rechercher les messages actuellement mis en file d'attente pour un abonnement, procédez comme suit:

Procédure

1. Pour rechercher les messages mis en file d'attente pour un type d'abonnement **DISPLAY SBSTATUS(<sub_name>) NUMMSGS**, voir «Affichage des attributs des abonnements», à la page 98.
2. Si la valeur **NUMMSGS** est supérieure à zéro, identifiez la file d'attente associée à l'abonnement en entrant **DISPLAY SUB(<sub_name>)DEST**.
3. En utilisant le nom de la file d'attente renvoyée, vous pouvez afficher les messages en suivant la technique décrite dans «Exploration des files d'attente», à la page 89.

Utilisation des services

Les objets de service sont un moyen par lequel des processus supplémentaires peuvent être gérés dans le cadre d'un gestionnaire de files d'attente. Avec les services, vous pouvez définir des programmes qui sont démarrés et arrêtés lorsque le gestionnaire de files d'attente démarre et se termine. Les services IBM WebSphere MQ sont toujours démarrés sous l'ID de l'utilisateur qui a démarré le gestionnaire de files d'attente.

Les objets de service peuvent être de l'un des types suivants:

serveur
Un serveur est un objet de service dont le paramètre SERVTYPE est défini comme SERVER. Un objet de service serveur est la définition d'un programme qui est exécuté lorsqu'un gestionnaire de files d'attente spécifié est démarré. Les objets de service serveur définissent des programmes qui s'exécutent généralement pendant une longue période. Par exemple, un objet de service de serveur peut être utilisé pour exécuter un processus de moniteur de déclenchement, tel que `runmqtrm`.

Une seule instance d'un objet de service de serveur peut s'exécuter simultanément. Le statut des objets de service du serveur en cours d'exécution peut être surveillé à l'aide de la commande MQSC, DISPLAY SVSTATUS.

Commande
Une commande est un objet de service dont le paramètre SERVTYPE est spécifié comme COMMAND. Les objets de service de commande sont similaires aux objets de service de serveur, mais plusieurs instances d'un objet de service de commande peuvent s'exécuter simultanément et leur statut ne peut pas être surveillé à l'aide de la commande MQSC DISPLAY SVSTATUS.

Si la commande MQSC, STOP SERVICE, est exécutée, aucune vérification n'est effectuée pour déterminer si le programme démarré par la commande MQSC, START SERVICE, est toujours actif avant l'exécution du programme d'arrêt.

Définition d'un objet de service

Vous définissez un objet de service avec différents attributs.

Les attributs sont les suivants:

SERVTYPE
Définit le type de l'objet de service. Les valeurs admises sont les suivantes :

SERVEUR
Un objet de service de serveur.

Une seule instance d'un objet de service de serveur peut être exécutée à la fois. Le statut des objets de service du serveur peut être surveillé à l'aide de la commande MQSC, DISPLAY SVSTATUS.

Commande
Objet de service de commande.

Plusieurs instances d'un objet de service de commande peuvent être exécutées simultanément. L'état des objets de service de commande ne peut pas être surveillé.

STARTCMD
Programme exécuté pour démarrer le service. Vous devez indiquer un chemin d'accès complet au programme.

STARTARG
Arguments transmis au programme de démarrage.

STDERR
Indique le chemin d'accès à un fichier vers lequel l'erreur standard (stderr) du programme de service doit être redirigée.

STDOUT
Indique le chemin d'accès à un fichier vers lequel la sortie standard (stdout) du programme de service doit être redirigée.

STOPCMD
Programme exécuté pour arrêter le service. Vous devez indiquer un chemin d'accès complet au programme.

STOPARG
Arguments transmis au programme d'arrêt.

CONTROL
Indique comment le service doit être démarré et arrêté:

MANUAL
Le service ne doit pas être démarré automatiquement ou arrêté automatiquement. Il est contrôlé par l'utilisation des commandes START SERVICE et STOP SERVICE. Il s'agit de la valeur par défaut.

QMGR
Le service défini doit être démarré et arrêté en même temps que le gestionnaire de files d'attente.

STARTONLY (UNIQUEMENT)
Le service doit être démarré en même temps que le gestionnaire de files d'attente, mais il n'est pas demandé de s'arrêter lorsque le gestionnaire de files d'attente est arrêté.

Concepts associés

«Gestion des services», à la page 101
A l'aide du paramètre CONTROL, une instance d'un objet de service peut être démarrée et arrêtée automatiquement par le gestionnaire de files d'attente, ou démarrée et arrêtée à l'aide des commandes MQSC START SERVICE et STOP SERVICE.

Gestion des services

A l'aide du paramètre CONTROL, une instance d'un objet de service peut être démarrée et arrêtée automatiquement par le gestionnaire de files d'attente, ou démarrée et arrêtée à l'aide des commandes MQSC START SERVICE et STOP SERVICE.

Lorsqu'une instance d'un objet de service est démarrée, un message contenant le nom de l'objet de service et l'ID de processus du processus démarré est consigné dans le journal des erreurs du gestionnaire de files d'attente. Voici un exemple d'entrée de journal pour le démarrage d'un objet de service de serveur:

```
  02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

  EXPLANATION:
  The Server process has started.
```

```
  ACTION:
  None.
```

Voici un exemple d'entrée de journal pour le démarrage d'un objet de service de commande:

```
  02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

  EXPLANATION:
  The Command has started.
  ACTION:
  None.
```

Lorsqu'un service de serveur d'instance s'arrête, un message est consigné dans les journaux d'erreurs du gestionnaire de files d'attente contenant le nom du service et l'ID de processus du processus en cours de fin. Voici un exemple d'entrée de journal pour l'arrêt d'un objet de service de serveur:

```
  02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

  EXPLANATION:
  The Server process has ended.
  ACTION:
  None.
```

Référence associée

«Autres variables d'environnement», à la page 102
L'environnement dans lequel le processus d'un service est démarré est hérité de celui du gestionnaire de files d'attente. Il est possible de définir des variables d'environnement supplémentaires à définir dans l'environnement du processus de service en ajoutant les variables à définir à l'un des fichiers de substitution d'environnement `service.env` .

Autres variables d'environnement

L'environnement dans lequel le processus d'un service est démarré est hérité de celui du gestionnaire de files d'attente. Il est possible de définir des variables d'environnement supplémentaires à définir dans l'environnement du processus de service en ajoutant les variables à définir à l'un des fichiers de substitution d'environnement `service.env` .

Remarque :

Il existe deux fichiers possibles auxquels vous pouvez ajouter des variables d'environnement:

- Le fichier `service.env` de portée machine, qui se trouve dans `/var/mqm` sur les systèmes UNIX and Linux ou dans le répertoire de données sélectionné lors de l'installation sur les systèmes Windows .
- Le fichier `service.env` de portée du gestionnaire de files d'attente, qui se trouve dans le répertoire de données du gestionnaire de files d'attente. Par exemple, l'emplacement du fichier de substitution d'environnement pour un gestionnaire de files d'attente nommé QMNAME est:
  - Sur les systèmes UNIX and Linux , `/var/mqm/qmgrs/QMNAME/service.env`
  - Sur les systèmes Windows , `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`

Les deux fichiers sont traités, s'ils sont disponibles, avec les définitions du fichier de portée du gestionnaire de files d'attente prioritaires sur celles du fichier de portée de la machine.

Toute variable d'environnement peut être spécifiée dans `service.env`. Par exemple, si le service IBM WebSphere MQ exécute un certain nombre de commandes, il peut être utile de définir la variable utilisateur PATH dans le fichier `service.env` . Les valeurs que vous définissez pour la variable ne peuvent pas être des variables d'environnement ; par exemple, CLASSPATH=*%CLASSPATH%* est incorrect. De même, sur Linux PATH=*$PATH*`:/opt/mqm/bin` , des résultats inattendus sont générés.

CLASSPATH doit être en majuscules et l'instruction de chemin d'accès aux classes ne peut contenir que des littéraux. Certains services (télémétrie, par exemple) définissent leur propre chemin d'accès aux classes. La variable CLASSPATH définie dans le fichier `service.env` est ajoutée à ce dernier.

Le format des variables définies dans le fichier`service.env` est une liste de paires de variables de nom et de valeur. Chaque variable doit être définie sur une nouvelle ligne, et chaque variable est prise telle qu'elle est explicitement définie, y compris les espaces. Voici un exemple de fichier,`service.env` :

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2024.                        *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                  *#
#*                                                                  *#
#********************************************************************#
#********************************************************************#
#* Module Name: service.env                                         *#
#* Type       : WebSphere MQ service environment file               *#
#* Function   : Define additional environment variables to be set   *#
#*              for SERVICE programs.                               *#
#* Usage      : <VARIABLE>=<VALUE>                                  *#
#*                                                                  *#
#********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

Référence associée

«Insertions remplaçables dans les définitions de service», à la page 103
Dans la définition d'un objet de service, il est possible de remplacer des jetons. Les jetons qui sont remplacés sont automatiquement remplacés par leur texte développé lors de l'exécution du programme de service. Les jetons de substitution peuvent être extraits de la liste suivante de jetons communs ou de toute variable définie dans le fichier, service.env.

Insertions remplaçables dans les définitions de service

Dans la définition d'un objet de service, il est possible de remplacer des jetons. Les jetons qui sont remplacés sont automatiquement remplacés par leur texte développé lors de l'exécution du programme de service. Les jetons de substitution peuvent être extraits de la liste suivante de jetons communs ou de toute variable définie dans le fichier, service.env.

Les jetons communs suivants peuvent être utilisés pour remplacer des jetons dans la définition d'un objet de service:

CHEMIN_INSTALL_MQ
Emplacement où WebSphere MQ est installé.

MQ_DATA_PATH
Emplacement du répertoire de données WebSphere MQ :

- Sur les systèmes UNIX and Linux , l'emplacement du répertoire de données WebSphere MQ est `/var/mqm/`
- Sur les systèmes Windows , l'emplacement du répertoire de données WebSphere MQ est le répertoire de données sélectionné lors de l'installation de WebSphere MQ

QMNAME
Nom du gestionnaire de files d'attente en cours.

NOM_SERVICE_MQ
Nom du service.

PID SERVEUR_MQ
Ce jeton ne peut être utilisé que par les arguments STOPARG et STOPCMD.

Pour les objets de service serveur, ce jeton est remplacé par l'ID de processus du processus démarré par les arguments STARTCMD et STARTARG. Sinon, ce jeton est remplacé par 0.

CHEMIN_DATA_MQ_Q_MGR_
Emplacement du répertoire de données du gestionnaire de files d'attente.

MQ_Q_MGR_DATA_NAME
Nom transformé du gestionnaire de files d'attente. Pour plus d'informations sur la transformation de nom, voir Présentation des noms de fichier WebSphere MQ.

Pour utiliser des insertions remplaçables, insérez le jeton dans les caractères + dans les chaînes STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT ou STDERR. Pour des exemples, voir «Exemples d'utilisation d'objets de service», à la page 104.

Exemples d'utilisation d'objets de service

Les services de cette section sont écrits avec des caractères de séparation de chemin de style UNIX , sauf indication contraire.

Utilisation d'un objet de service de serveur

Cet exemple montre comment définir, utiliser et modifier un objet de service de serveur pour démarrer un moniteur de déclenchement.

1. Un objet de service de serveur est défini à l'aide de la commande MQSC suivante:

```
DEFINE SERVICE(S1) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
       STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
       STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
       STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

   où :

   +MQ_INSTALL_PATH+ est un jeton représentant le répertoire d'installation.
   +QMNAME+ est un jeton représentant le nom du gestionnaire de files d'attente.
   ACCOUNTS.INITIATION.QUEUE est la file d'attente d'initialisation.
   amqsstop est un exemple de programme fourni avec WebSphere MQ qui demande au gestionnaire de files d'attente d'interrompre toutes les connexions pour l'ID de processus. amqsstop génère des commandes PCF. Par conséquent, le serveur de commandes doit être en cours d'exécution.
   +MQ_SERVER_PID+ est un jeton représentant l'ID de processus transmis au programme d'arrêt.

   Pour obtenir la liste des jetons communs, voir «Insertions remplaçables dans les définitions de service», à la page 103 .

2. Une instance de l'objet de service du serveur s'exécute lors du prochain démarrage du gestionnaire de files d'attente. Toutefois, nous démarrons immédiatement une instance de l'objet de service serveur à l'aide de la commande MQSC suivante:

```
START SERVICE(S1)
```

3. Le statut du processus de service du serveur est affiché à l'aide de la commande MQSC suivante:

```
DISPLAY SVSTATUS(S1)
```

4. Cet exemple montre maintenant comment modifier l'objet de service du serveur et faire en sorte que les mises à jour soient prises en compte en redémarrant manuellement le processus de service du

serveur. L'objet de service du serveur est modifié de sorte que la file d'attente d'initialisation soit spécifiée sous la forme `JUPITER.INITIATION.QUEUE` . La commande MQSC suivante est utilisée:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

Remarque : Un service en cours d'exécution ne récupère aucune mise à jour de sa définition de service tant qu'il n'est pas redémarré.

5. Le processus de service du serveur est redémarré afin que la modification soit prise en compte, à l'aide des commandes MQSC suivantes:

```
STOP SERVICE(S1)
```

suivie de :

```
START SERVICE(S1)
```

Le processus de service du serveur est redémarré et prend en compte les modifications apportées dans «4», à la page 104.

Remarque : La commande MQSC, STOP SERVICE, ne peut être utilisée que si un argument STOPCMD est spécifié dans la définition de service.

Utilisation d'un objet de service de commande

Cet exemple montre comment définir un objet de service de commande pour démarrer un programme qui écrit des entrées dans le journal système du système d'exploitation lorsqu'un gestionnaire de files d'attente est démarré ou arrêté.

1. L'objet de service de commande est défini à l'aide de la commande MQSC suivante:

```
DEFINE SERVICE(S2) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STARTCMD('/usr/bin/logger') +
       STARTARG('Queue manager +QMNAME+ starting') +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

où :

`logger` est la commande fournie par le système UNIX and Linux pour écrire dans le journal système.
+QMNAME+ est un jeton représentant le nom du gestionnaire de files d'attente.

Utilisation d'un objet de service de commande lorsqu'un gestionnaire de files d'attente se termine uniquement

Cet exemple montre comment définir un objet de service de commande pour démarrer un programme qui écrit des entrées dans le journal système du système d'exploitation lorsqu'un gestionnaire de files d'attente est arrêté uniquement.

1. L'objet de service de commande est défini à l'aide de la commande MQSC suivante:

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

où :

`logger` est un exemple de programme fourni avec WebSphere MQ qui peut écrire des entrées dans le journal système du système d'exploitation.

+QMNAME+ est un jeton représentant le nom du gestionnaire de files d'attente.

Plus sur la transmission d'arguments

Cet exemple montre comment définir un objet de service de serveur pour démarrer un programme appelé `runserv` lorsqu'un gestionnaire de files d'attente est démarré.

Cet exemple est écrit avec des caractères de séparation de chemin de style Windows .

L'un des arguments à transmettre au programme de démarrage est une chaîne contenant un espace. Cet argument doit être transmis sous la forme d'une chaîne unique. Pour ce faire, des guillemets sont utilisés comme indiqué dans la commande suivante pour définir l'objet de service de commande:

1. L'objet de service serveur est défini à l'aide de la commande MQSC suivante:

```
   DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
     STARTCMD('C:\Program Files\Tools\runserv.exe') +
     STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
     STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')


DEFINE SERVICE(S4) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('C:\Program Files\Tools\runserv.exe') +
       STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
       STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
```

où :

+QMNAME+ est un jeton représentant le nom du gestionnaire de files d'attente.

`"C:\Program Files\Tools\"` est une chaîne contenant un espace, qui sera transmise sous la forme d'une chaîne unique.

Démarrage automatique d'un service

Cet exemple montre comment définir un objet de service de serveur qui peut être utilisé pour démarrer automatiquement le moniteur de déclenchement lorsque le gestionnaire de files d'attente démarre.

1. L'objet de service serveur est défini à l'aide de la commande MQSC suivante:

```
DEFINE SERVICE(TRIG_MON_START) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('runmqtrm') +
       STARTARG('-m +QMNAME+ -q +IQNAME+')
```

où :

+QMNAME+ est un jeton représentant le nom du gestionnaire de files d'attente.

+IQNAME+ est une variable d'environnement définie par l'utilisateur dans l'un des fichiers service.env représentant le nom de la file d'attente d'initialisation.

Gestion des objets pour le déclenchement

WebSphere MQ vous permet de démarrer automatiquement une application lorsque certaines conditions d'une file d'attente sont remplies. Par exemple, vous pouvez démarrer une application lorsque le nombre de messages d'une file d'attente atteint un nombre spécifié. Cette fonction est appelée *déclenchement*. Vous devez définir les objets qui prennent en charge le déclenchement.

Déclenchement décrit en détail dans Démarrage d'applications WebSphere MQ à l'aide de déclencheurs.

Définition d'une file d'attente d'application pour le déclenchement

Une file d'attente d'application est une file d'attente locale utilisée par les applications pour la messagerie, via l'interface MQI. Le déclenchement requiert la définition d'un certain nombre d'attributs de file d'attente dans la file d'attente d'application.

Le déclenchement lui-même est activé par l'attribut *Trigger* (TRIGGER dans les commandes MQSC). Dans cet exemple, un événement déclencheur doit être généré lorsqu'il y a 100 messages de priorité 5 ou supérieure dans la file d'attente locale MOTOR.INSURANCE.QUEUE, comme suit:

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
       PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
       MAXMSGL (2000) +
       DEFPSIST (YES) +
       INITQ (MOTOR.INS.INIT.QUEUE) +
       TRIGGER +
       TRIGTYPE (DEPTH) +
       TRIGDPTH (100)+
       TRIGMPRI (5)
```

où :

QLOCAL (MOTOR.INSURANCE.QUEUE)
Nom de la file d'attente d'application en cours de définition.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
Nom de la définition de processus qui définit l'application à démarrer par un programme de moniteur de déclenchement.

MAXMSGL (2000)
Longueur maximale des messages dans la file d'attente.

DEFPSIST (YES)
Indique que les messages de cette file d'attente sont persistants par défaut.

INITQ (MOTOR.INS.INIT.QUEUE)
Nom de la file d'attente d'initialisation dans laquelle le gestionnaire de files d'attente doit insérer le message de déclenchement.

TRIGGER
Valeur de l'attribut de déclencheur.

TRIGTYPE (DEPTH)
Indique qu'un événement déclencheur est généré lorsque le nombre de messages de priorité requise (TRIGMPRI) atteint le nombre indiqué dans TRIGDPTH.

TRIGDPTH (100)
Nombre de messages requis pour générer un événement déclencheur.

TRIGMPRI (5)
Priorité des messages qui doivent être comptés par le gestionnaire de files d'attente pour déterminer s'il convient de générer un événement déclencheur. Seuls les messages de priorité 5 ou supérieure sont comptés.

Définition d'une file d'attente d'initialisation

Lorsqu'un événement déclencheur se produit, le gestionnaire de files d'attente place un message de déclenchement dans la file d'attente d'initialisation spécifiée dans la définition de file d'attente d'application. Les files d'attente d'initialisation n'ont pas de paramètres spéciaux, mais vous pouvez utiliser la définition suivante de la file d'attente locale MOTOR.INS.INIT.QUEUE pour obtenir des conseils:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Définition d'un processus

Utilisez la commande DEFINE PROCESS pour créer une définition de processus. Une définition de processus définit l'application à utiliser pour traiter les messages de la file d'attente d'application. La définition de file d'attente d'application nomme le processus à utiliser et associe ainsi la file d'attente d'application à l'application à utiliser pour traiter ses messages. Cette opération est effectuée via l'attribut PROCESS de la file d'attente d'application MOTOR.INSURANCE.QUEUE. La commande MQSC suivante définit le processus requis, MOTOR.INSURANCE.QUOTE.PROCESS, identifié dans cet exemple:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

où :

MOTOR.INSURANCE.QUOTE.PROCESS
Nom de la définition de processus.

DESCR ('Insurance request message processing')
Décrit le programme d'application auquel cette définition se rapporte. Ce texte s'affiche lorsque vous utilisez la commande DISPLAY PROCESS. Cela peut vous aider à identifier ce que le processus fait. Si vous utilisez des espaces dans la chaîne, vous devez placer la chaîne entre apostrophes.

APPLTYPE (UNIX)
Type d'application à démarrer.

APPLICID ('/u/admin/test/IRMP01')
Nom du fichier exécutable de l'application, indiqué comme nom de fichier qualifié complet. Sur les systèmes Windows , une valeur APPLICID typique serait `c:\appl\test\irmp01.exe`.

USERDATA ('open, close, 235')
Données définies par l'utilisateur qui peuvent être utilisées par l'application.

Affichage des attributs d'une définition de processus

Utilisez la commande DISPLAY PROCESS pour examiner les résultats de votre définition. Exemple :

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


    24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
    DESCR ('Insurance request message processing')
    APPLICID ('/u/admin/test/IRMP01')
    USERDATA (open, close, 235)
    PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
    APPLTYPE (UNIX)
```

Vous pouvez également utiliser la commande MQSC ALTER PROCESS pour modifier une définition de processus existante et la commande DELETE PROCESS pour supprimer une définition de processus.

Administration des objets IBM WebSphere MQ distants

Cette section explique comment administrer des objets IBM WebSphere MQ sur un gestionnaire de files d'attente éloignées à l'aide de commandes MQSC et comment utiliser des objets de file d'attente éloignée pour contrôler la destination des messages et des messages de réponse.

Cette section décrit :

- «Canaux, clusters et mise en file d'attente distante», à la page 109
- «Administration à distance à partir d'un gestionnaire de files d'attente local», à la page 110
- «Création d'une définition locale d'une file d'attente éloignée», à la page 116

- «Utilisation de définitions de file d'attente éloignée comme alias», à la page 119
- «Conversion de données entre des jeux de caractères codés», à la page 120

Canaux, clusters et mise en file d'attente distante

Un gestionnaire de files d'attente communique avec un autre gestionnaire de files d'attente en envoyant un message et, si nécessaire, en recevant une réponse. Le gestionnaire de files d'attente de réception peut être:

- Sur la même machine
- Sur une autre machine au même endroit (ou même de l'autre côté du monde)
- Exécution sur la même plateforme que le gestionnaire de files d'attente local
- Exécution sur une autre plateforme prise en charge par WebSphere MQ

Ces messages peuvent provenir de:

- Programmes d'application écrits par l'utilisateur qui transfèrent des données d'un noeud à un autre
- Applications d'administration écrites par l'utilisateur qui utilisent des commandes PCF ou MQAI
- L'explorateur IBM WebSphere MQ .
- Envoi des gestionnaires de files d'attente:
  - Messages d'événement d'instrumentation vers un autre gestionnaire de files d'attente
  - Commandes MQSC émises à partir d'une commande `runmqsc` en mode indirect (où les commandes sont exécutées sur un autre gestionnaire de files d'attente)

Avant qu'un message puisse être envoyé à un gestionnaire de files d'attente éloignées, le gestionnaire de files d'attente local a besoin d'un mécanisme permettant de détecter l'arrivée des messages et de les transporter, à savoir:

- Au moins un canal
- Une file d'attente de transmission
- Un initiateur de canal

Pour qu'un gestionnaire de files d'attente éloignées reçoive un message, un programme d'écoute est requis.

Un canal est une liaison de communication unidirectionnelle entre deux gestionnaires de files d'attente et peut transporter des messages destinés à n'importe quel nombre de files d'attente sur le gestionnaire de files d'attente éloignées.

Chaque extrémité du canal possède une définition distincte. Par exemple, si une extrémité est un expéditeur ou un serveur, l'autre extrémité doit être un destinataire ou un demandeur. Un canal simple se compose d'une *définition de canal émetteur* à l'extrémité du gestionnaire de files d'attente local et d'une *définition de canal récepteur* à l'extrémité du gestionnaire de files d'attente éloigné. Les deux définitions doivent avoir le même nom et constituer ensemble un canal de transmission de messages unique.

Si vous souhaitez que le gestionnaire de files d'attente éloignées réponde aux messages envoyés par le gestionnaire de files d'attente local, configurez un deuxième canal pour renvoyer les réponses au gestionnaire de files d'attente local.

Utilisez la commande MQSC DEFINE CHANNEL pour définir des canaux. Dans cette section, les exemples relatifs aux canaux utilisent les attributs de canal par défaut, sauf indication contraire.

Il existe un agent MCA à chaque extrémité d'un canal, qui contrôle l'envoi et la réception des messages. L'agent MCA extrait les messages de la file d'attente de transmission et les place sur la liaison de communication entre les gestionnaires de files d'attente.

Une file d'attente de transmission est une file d'attente locale spécialisée qui contient temporairement des messages avant que l'agent MCA ne les récupère et les envoie au gestionnaire de files d'attente

éloignées. Vous spécifiez le nom de la file d'attente de transmission dans une *définition de file d'attente éloignée*.

Vous pouvez autoriser un agent MCA à transférer des messages via plusieurs unités d'exécution. Ce processus est connu sous le nom de *principe du pipeline*. Le principe du pipeline permet à l'agent MCA de transférer des messages plus efficacement, améliorant ainsi les performances de canal. Pour plus d'informations sur la configuration d'un canal en vue de l'utilisation du pipeline, voir Attributs des canaux .

«Préparation des canaux et des files d'attente de transmission pour l'administration à distance», à la page 111 vous explique comment utiliser ces définitions pour configurer l'administration à distance.

Pour plus d'informations sur la configuration de la mise en file d'attente répartie en général, voir Composants de la mise en file d'attente répartie.

Administration à distance à l'aide de clusters

Dans un réseau WebSphere MQ utilisant la mise en file d'attente répartie, chaque gestionnaire de files d'attente est indépendant. Si un gestionnaire de files d'attente doit envoyer des messages à un autre gestionnaire de files d'attente, il doit définir une file d'attente de transmission, un canal vers le gestionnaire de files d'attente éloignées et une définition de file d'attente éloignée pour chaque file d'attente à laquelle il souhaite envoyer des messages.

Un *cluster* est un groupe de gestionnaires de files d'attente configurés de telle sorte que les gestionnaires de files d'attente puissent communiquer directement entre eux sur un réseau unique sans file d'attente de transmission, canal et définitions de file d'attente complexes. Les clusters peuvent être configurés facilement et contiennent généralement des gestionnaires de files d'attente qui sont logiquement liés d'une manière ou d'une autre et qui doivent partager des données ou des applications. Même le plus petit cluster réduit les coûts d'administration du système.

L'établissement d'un réseau de gestionnaires de files d'attente dans un cluster implique moins de définitions que l'établissement d'un environnement de mise en file d'attente répartie traditionnel. Avec moins de définitions à créer, vous pouvez configurer ou modifier votre réseau plus rapidement et plus facilement, et réduire le risque d'erreur dans vos définitions.

Pour configurer un cluster, vous avez besoin d'une définition d'émetteur de cluster (CLUSSDR) et d'un récepteur de cluster (CLUSRCVR) pour chaque gestionnaire de files d'attente. Vous n'avez pas besoin de définitions de file d'attente de transmission ni de définitions de file d'attente éloignée. Les principes de l'administration à distance sont les mêmes lorsqu'ils sont utilisés dans un cluster, mais les définitions elles-mêmes sont grandement simplifiées.

Pour plus d'informations sur les clusters, leurs attributs et leur configuration, voir Clusters de gestionnaires de files d'attente.

Administration à distance à partir d'un gestionnaire de files d'attente local

Cette section explique comment administrer un gestionnaire de files d'attente éloignées à partir d'un gestionnaire de files d'attente local à l'aide des commandes MQSC et PCF.

La préparation des files d'attente et des canaux est essentiellement la même pour les commandes MQSC et PCF. Dans cette section, les exemples montrent des commandes MQSC, car elles sont plus faciles à comprendre. Pour plus d'informations sur l'écriture de programmes d'administration à l'aide de commandes PCF, voir «Utilisation des formats de commande programmables», à la page 10.

Vous envoyez des commandes MQSC à un gestionnaire de files d'attente éloignées de manière interactive ou à partir d'un fichier texte contenant les commandes. Le gestionnaire de files d'attente éloignées peut se trouver sur la même machine ou, plus généralement, sur une machine différente. Vous pouvez administrer à distance des gestionnaires de files d'attente dans d'autres environnements WebSphere MQ , y compris les systèmes UNIX and Linux , les systèmes Windows , IBM iet z/OS.

Pour implémenter l'administration à distance, vous devez créer des objets spécifiques. Sauf si vous avez des exigences spécifiques, les valeurs par défaut (par exemple, pour la longueur maximale des messages) sont suffisantes.

Préparation des gestionnaires de files d'attente pour l'administration à distance

Comment utiliser les commandes MQSC pour préparer les gestionnaires de files d'attente pour l'administration à distance.

La Figure 17, à la page 111 présente la configuration des gestionnaires de files d'attente et des canaux dont vous avez besoin pour l'administration à distance à l'aide de la commande **runmqsc** . L'objet `source.queue.manager` est le gestionnaire de files d'attente source à partir duquel vous pouvez émettre des commandes MQSC et auquel les résultats de ces commandes (messages de l'opérateur) sont renvoyés. L'objet `target.queue.manager` est le nom du gestionnaire de files d'attente cible, qui traite les commandes et génère des messages d'opérateur.

Remarque : Si vous utilisez **runmqsc** avec l'option -w, `source.queue.manager` ***doit*** être le gestionnaire de files d'attente par défaut. Pour plus d'informations sur la création d'un gestionnaire de files d'attente, voir crtmqm.

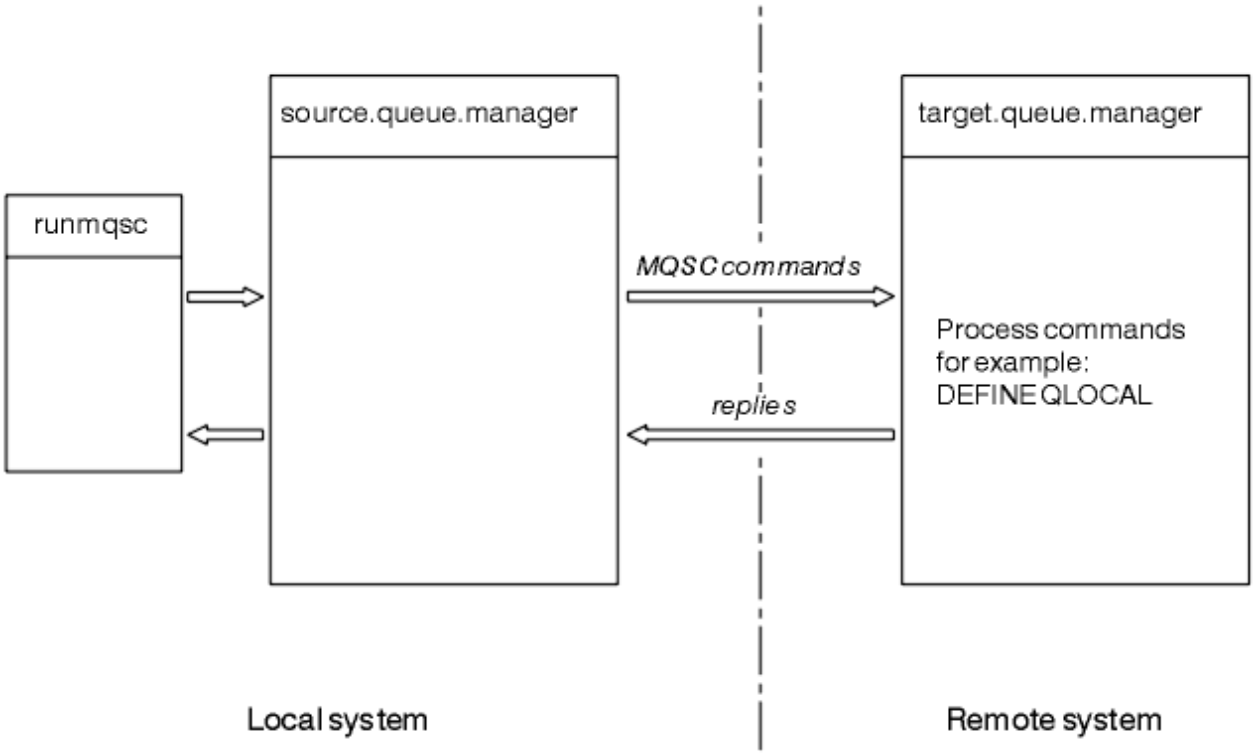


Figure 17. Administration à distance à l'aide des commandes MQSC

Sur les deux systèmes, si vous ne l'avez pas déjà fait:

- Créez le gestionnaire de files d'attente et les objets par défaut à l'aide de la commande `crtmqm` .
- Démarrez le gestionnaire de files d'attente à l'aide de la commande `strmqm` .

Sur le gestionnaire de files d'attente cible:

- La file d'attente de commandes, SYSTEM.ADMIN.COMMAND.QUEUE, doit être présent. Cette file d'attente est créée par défaut lorsqu'un gestionnaire de files d'attente est créé.

Vous devez exécuter ces commandes localement ou via une fonction réseau telle que Telnet.

Préparation des canaux et des files d'attente de transmission pour l'administration à distance

Comment utiliser les commandes MQSC pour préparer les canaux et les files d'attente de transmission pour l'administration à distance.

Pour exécuter des commandes MQSC à distance, configurez deux canaux, un pour chaque direction et leurs files d'attente de transmission associées. Cet exemple suppose que vous utilisez TCP/IP comme type de transport et que vous connaissez l'adresse TCP/IP impliquée.

Le canal `source.to.target` permet d'envoyer des commandes MQSC du gestionnaire de files d'attente source vers le gestionnaire de files d'attente cible. Son émetteur est `source.queue.manager` et son récepteur est `target.queue.manager`. Le canal `target.to.source` permet de renvoyer la sortie des commandes et des messages de l'opérateur générés vers le gestionnaire de files d'attente source. Vous devez également définir une file d'attente de transmission pour chaque canal. Cette file d'attente est une file d'attente locale à laquelle est attribué le nom du gestionnaire de files d'attente de réception. Le nom XMITQ doit correspondre au nom du gestionnaire de files d'attente éloignées pour que l'administration à distance fonctionne, sauf si vous utilisez un alias de gestionnaire de files d'attente. Figure 18, à la page 112 résume cette configuration.

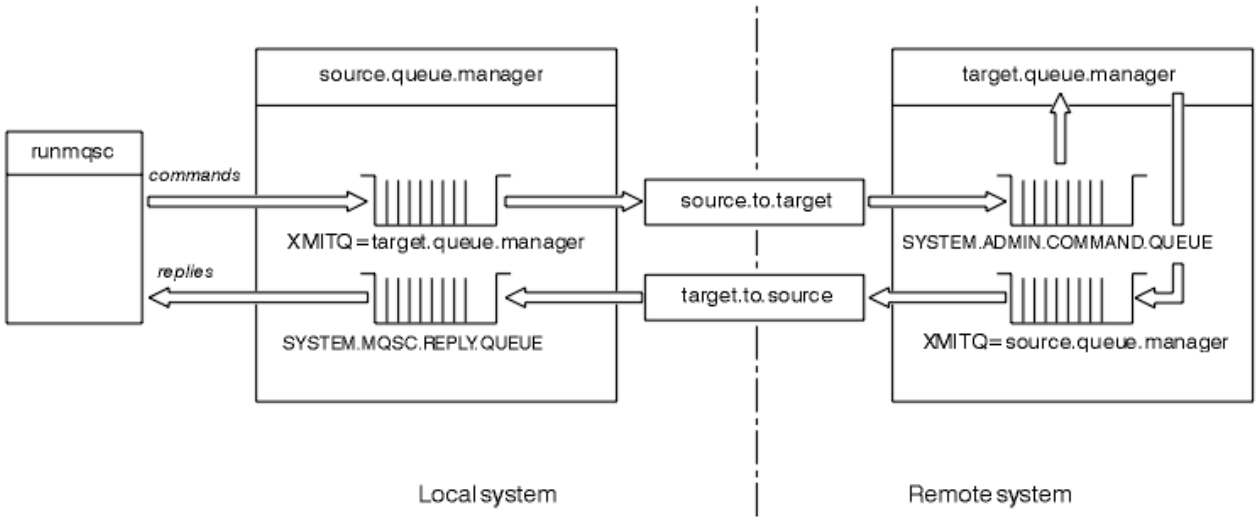


Figure 18. Configuration de canaux et de files d'attente pour l'administration à distance

Pour plus d'informations sur la configuration des canaux, voir Connexion d'applications à l'aide de la mise en file d'attente répartie .

Définition des canaux, des programmes d'écoute et des files d'attente de transmission

Sur le gestionnaire de files d'attente source (`source.queue.manager`), exécutez les commandes MQSC suivantes pour définir les canaux, le programme d'écoute et la file d'attente de transmission:

1. Définissez le canal émetteur sur le gestionnaire de files d'attente source:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. Définissez le canal récepteur sur le gestionnaire de files d'attente source:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Définissez le programme d'écoute sur le gestionnaire de files d'attente source:

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. Définissez la file d'attente de transmission sur le gestionnaire de files d'attente source:

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

Exécutez les commandes suivantes sur le gestionnaire de files d'attente cible (`target.queue.manager`) pour créer les canaux, le programme d'écoute et la file d'attente de transmission:

1. Définissez le canal émetteur sur le gestionnaire de files d'attente cible:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. Définissez le canal récepteur sur le gestionnaire de files d'attente cible:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Définissez le programme d'écoute sur le gestionnaire de files d'attente cible:

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. Définissez la file d'attente de transmission sur le gestionnaire de files d'attente cible:

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

Remarque : Les noms de connexion TCP/IP indiqués pour l'attribut CONNAME dans les définitions de canal émetteur ne sont donnés qu'à titre d'illustration. Il s'agit du nom de réseau de la machine à l' *autre* extrémité de la connexion. Utilisez les valeurs appropriées pour votre réseau.

Démarrage des programmes d'écoute et des canaux

Comment utiliser les commandes MQSC pour démarrer les programmes d'écoute et les canaux.

Démarrez les deux programmes d'écoute à l'aide des commandes MQSC suivantes:

1. Démarrez le programme d'écoute sur le gestionnaire de files d'attente source, `source.queue.manager`, en exécutant la commande MQSC suivante:

```
START LISTENER ('source.queue.manager')
```

2. Démarrez le programme d'écoute sur le gestionnaire de files d'attente cible, `target.queue.manager`, en exécutant la commande MQSC suivante:

```
START LISTENER ('target.queue.manager')
```

Démarrez les deux canaux émetteurs à l'aide des commandes MQSC suivantes:

1. Démarrez le canal émetteur sur le gestionnaire de files d'attente source, `source.queue.manager`, en émettant la commande MQSC suivante:

```
START CHANNEL ('source.to.target')
```

2. Démarrez le canal émetteur sur le gestionnaire de files d'attente cible, `target.queue.manager`, en exécutant la commande MQSC suivante:

```
START CHANNEL ('target.to.source')
```

Définition automatique des canaux
Vous activez la définition automatique des définitions de récepteur et de connexion serveur en mettant à jour l'objet gestionnaire de files d'attente à l'aide de la commande MQSC ALTER QMGR (ou de la commande PCF Change Queue Manager).

Si WebSphere MQ reçoit une demande de connexion entrante et ne parvient pas à trouver un canal de réception ou de connexion serveur approprié, il crée automatiquement un canal. Les définitions automatiques sont basées sur deux définitions par défaut fournies avec WebSphere MQ: SYSTEM.AUTO.RECEIVER et SYSTEM.AUTO.SVRCONN.

Pour plus d'informations sur la création automatique de définitions de canal, voir Préparation des canaux. Pour plus d'informations sur la définition automatique des canaux pour les clusters, voir Définition automatique des canaux de cluster.

Gestion du serveur de commandes pour l'administration à distance

Comment démarrer, arrêter et afficher le statut du serveur de commandes. Un serveur de commandes est obligatoire pour toute administration impliquant des commandes PCF, MQAI, ainsi que pour l'administration à distance.

Chaque gestionnaire de files d'attente peut être associé à un serveur de commandes. Un serveur de commandes traite toutes les commandes entrantes provenant des gestionnaires de files d'attente éloignées ou les commandes PCF provenant des applications. Il présente les commandes au gestionnaire de files d'attente pour le traitement et renvoie un code achèvement ou un message d'opérateur en fonction de l'origine de la commande.

Remarque : Pour l'administration à distance, vérifiez que le gestionnaire de files d'attente cible est en cours d'exécution. Sinon, les messages contenant des commandes ne peuvent pas quitter le gestionnaire de files d'attente à partir duquel elles sont émises. A la place, ces messages sont mis en file d'attente dans la file d'attente de transmission locale qui sert le gestionnaire de files d'attente éloignées. Evitez cette situation.

Il existe des commandes de contrôle distinctes pour le démarrage et l'arrêt du serveur de commandes. A condition que le serveur de commandes soit en cours d'exécution, les utilisateurs de WebSphere MQ for Windows ou WebSphere MQ for Linux (plateformesx86 et x86-64) peuvent effectuer les opérations décrites dans les sections suivantes à l'aide de l'explorateur IBM WebSphere MQ . Pour plus d'informations, voir «Administration à l'aide de IBM WebSphere MQ Explorer», à la page 58.

Démarrage du serveur de commandes

Selon la valeur de l'attribut de gestionnaire de files d'attente, *SCMDSERV*, le serveur de commandes est démarré automatiquement au démarrage du gestionnaire de files d'attente ou doit être démarré manuellement. La valeur de l'attribut de gestionnaire de files d'attente peut être modifiée à l'aide de la commande MQSC ALTER QMGR spécifiant le paramètre SCMDSERV. Par défaut, le serveur de commandes est démarré automatiquement.

Si *SCMDSERV* est défini sur MANUAL, démarrez le serveur de commandes à l'aide de la commande suivante:

```
strmqcsv saturn.queue.manager
```

où `saturn.queue.manager` est le gestionnaire de files d'attente pour lequel le serveur de commandes est démarré.

Affichage de l'état du serveur de commandes

Pour l'administration à distance, vérifiez que le serveur de commandes sur le gestionnaire de files d'attente cible est en cours d'exécution. S'il n'est pas en cours d'exécution, les commandes distantes ne peuvent pas être traitées. Tous les messages contenant des commandes sont mis en file d'attente dans la file d'attente de commandes du gestionnaire de files d'attente cible.

Pour afficher le statut du serveur de commandes d'un gestionnaire de files d'attente, exécutez la commande MQSC suivante:

```
DISPLAY QMSTATUS CMDSERV
```

Arrêt d'un serveur de commandes

Pour arrêter le serveur de commandes démarré par l'exemple précédent, utilisez la commande suivante:

```
endmqcsv saturn.queue.manager
```

Vous pouvez arrêter le serveur de commandes de deux manières:

- Pour un arrêt contrôlé, utilisez la commande `endmqcsv` avec l'indicateur -c, qui est la valeur par défaut.
- Pour un arrêt immédiat, utilisez la commande `endmqcsv` avec l'indicateur -i.

Remarque : L'arrêt d'un gestionnaire de files d'attente arrête également le serveur de commandes qui lui est associé.

Emission de commandes MQSC sur un gestionnaire de files d'attente éloignées

Vous pouvez utiliser une forme particulière de la commande **`runmqsc`** pour exécuter des commandes MQSC sur un gestionnaire de files d'attente éloignées.

Le serveur de commandes ***doit*** être en cours d'exécution sur le gestionnaire de files d'attente cible, s'il va traiter des commandes MQSC à distance. (Cette opération n'est pas nécessaire sur le gestionnaire de files d'attente source). Pour plus d'informations sur le démarrage du serveur de commandes sur un gestionnaire de files d'attente, voir «Gestion du serveur de commandes pour l'administration à distance», à la page 114.

Sur le gestionnaire de files d'attente source, vous pouvez ensuite exécuter les commandes MQSC de manière interactive en mode indirect en entrant:

```
runmqsc -w 30 target.queue.manager
```

Cette forme de la commande `runmqsc` , avec l'indicateur -w, exécute les commandes MQSC en mode indirect, où les commandes sont placées (sous une forme modifiée) dans la file d'attente d'entrée du serveur de commandes et exécutées dans l'ordre.

Lorsque vous entrez une commande MQSC, elle est redirigée vers le gestionnaire de files d'attente éloignées, dans ce cas, `target.queue.manager`. Le délai d'attente est de 30 secondes ; si une réponse n'est pas reçue dans les 30 secondes, le message suivant est généré sur le gestionnaire de files d'attente local (source):

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Lorsque vous arrêtez d'émettre des commandes MQSC, le gestionnaire de files d'attente local affiche toutes les réponses arrivées à expiration et supprime toutes les autres réponses.

Le gestionnaire de files d'attente source prend par défaut la valeur du gestionnaire de files d'attente local par défaut. Si vous spécifiez l'option `-m` *LocalQmgrNom* dans la commande **runmqsc** , vous pouvez indiquer les commandes à émettre par l'intermédiaire d'un gestionnaire de files d'attente local.

En mode indirect, vous pouvez également exécuter un fichier de commandes MQSC sur un gestionnaire de files d'attente éloignées. Exemple :

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

où `mycomds.in` est un fichier contenant les commandes MQSC et `report.out` est le fichier de rapport.

Méthode suggérée pour l'émission de commandes à distance

Lorsque vous émettez des commandes sur un gestionnaire de files d'attente éloignées, envisagez d'utiliser l'approche suivante:

1. Placez les commandes MQSC à exécuter sur le système distant dans un fichier de commandes.
2. Vérifiez vos commandes MQSC en local en spécifiant l'indicateur -v dans la commande `runmqsc` .

   Vous ne pouvez pas utiliser `runmqsc` pour vérifier les commandes MQSC sur un autre gestionnaire de files d'attente.
3. Vérifiez que le fichier de commandes s'exécute localement sans erreur.
4. Exécutez le fichier de commandes sur le système distant.

Si vous rencontrez des problèmes lors de l'utilisation à distance des commandes MQSC

Si vous avez des difficultés à exécuter des commandes MQSC à distance, vérifiez que vous disposez des éléments suivants:

- Démarrage du serveur de commandes sur le gestionnaire de files d'attente cible.
- Une file d'attente de transmission valide a été définie.
- Définissez les deux extrémités des canaux de transmission de messages pour les deux:
  - Canal via lequel les commandes sont envoyées.
  - Canal par lequel les réponses doivent être renvoyées.
- Indique le nom de connexion correct (CONNAME) dans la définition de canal.
- Vous avez démarré les programmes d'écoute avant de démarrer les canaux de transmission de messages.
- Vérification que l'intervalle de déconnexion n'a pas expiré, par exemple, si un canal a démarré, puis s'est arrêté après un certain temps. Ceci est particulièrement important si vous démarrez les canaux manuellement.
- Demandes envoyées à partir d'un gestionnaire de files d'attente source qui n'ont pas de sens pour le gestionnaire de files d'attente cible (par exemple, des demandes incluant des paramètres qui ne sont pas pris en charge sur le gestionnaire de files d'attente éloignées).

Voir aussi «Résolution des problèmes liés aux commandes MQSC», à la page 83.

Création d'une définition locale d'une file d'attente éloignée

Une définition locale d'une file d'attente éloignée est une définition sur un gestionnaire de files d'attente local qui fait référence à une file d'attente sur un gestionnaire de files d'attente éloignées.

Vous n'avez pas besoin de définir une file d'attente éloignée à partir d'une position locale, mais l'avantage est que les applications peuvent faire référence à la file d'attente éloignée par son nom défini localement au lieu d'avoir à spécifier un nom qualifié par l'ID du gestionnaire de files d'attente sur lequel se trouve la file d'attente éloignée.

Description du fonctionnement des définitions locales des files d'attente distantes

Une application se connecte à un gestionnaire de files d'attente local, puis émet un appel MQOPEN . Dans l'appel ouvert, le nom de file d'attente spécifié est celui d'une définition de file d'attente éloignée sur le gestionnaire de files d'attente local. La définition de file d'attente éloignée fournit les noms de la file d'attente cible, du gestionnaire de files d'attente cible et, éventuellement, d'une file d'attente de transmission. Pour placer un message dans la file d'attente éloignée, l'application émet un appel MQPUT en spécifiant le descripteur renvoyé par l'appel MQOPEN . Le gestionnaire de files d'attente utilise le nom de la file d'attente éloignée et le nom du gestionnaire de files d'attente éloignées dans un en-tête de transmission au début du message. Ces informations sont utilisées pour acheminer le message vers sa destination correcte dans le réseau.

En tant qu'administrateur, vous pouvez contrôler la destination du message en modifiant la définition de la file d'attente éloignée.

L'exemple suivant montre comment une application insère un message dans une file d'attente appartenant à un gestionnaire de files d'attente éloignées. L'application se connecte à un gestionnaire de files d'attente, par exemple, `saturn.queue.manager`. La file d'attente cible appartient à un autre gestionnaire de files d'attente.

Dans l'appel MQOPEN , l'application spécifie les zones suivantes:

Zone Valeur	Description
ObjectName CYAN.REMOTE.QUEUE	Indique le nom local de l'objet file d'attente éloignée. Définit la file d'attente cible et le gestionnaire de files d'attente cible.
ObjectType (File d'attente)	Identifie cet objet en tant que file d'attente.
ObjectQmgrName Vide ou `saturn.queue.manager`	Cette zone est facultative. Si cette zone est vide, le nom du gestionnaire de files d'attente local est pris en compte. (Il s'agit du gestionnaire de files d'attente sur lequel la définition de file d'attente éloignée existe.)

Après cela, l'application émet un appel MQPUT pour insérer un message dans cette file d'attente.

Sur le gestionnaire de files d'attente local, vous pouvez créer une définition locale d'une file d'attente éloignée à l'aide des commandes MQSC suivantes:

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
       DESCR ('Queue for auto insurance requests from the branches') +
       RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
       RQMNAME (jupiter.queue.manager) +
       XMITQ (INQUOTE.XMIT.QUEUE)
```

où :

QREMOTE (CYAN.REMOTE.QUEUE)
Indique le nom local de l'objet file d'attente éloignée. Il s'agit du nom que les applications connectées à ce gestionnaire de files d'attente doivent spécifier dans l'appel MQOPEN pour ouvrir la file d'attente AUTOMOBILE.INSURANCE.QUOTE.QUEUE sur le gestionnaire de files d'attente éloignées `jupiter.queue.manager`.

DESCR ('Queue for auto insurance requests from the branches')
Fournit un texte supplémentaire qui décrit l'utilisation de la file d'attente.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Indique le nom de la file d'attente cible sur le gestionnaire de files d'attente éloignées. Il s'agit de la file d'attente cible réelle pour les messages envoyés par les applications qui spécifient le nom de file d'attente CYAN.REMOTE.QUEUE. File d'attente AUTOMOBILE.INSURANCE.QUOTE.QUEUE doit être définie en tant que file d'attente locale sur le gestionnaire de files d'attente éloignées.

RQMNAME (jupiter.queue.manager)
Indique le nom du gestionnaire de files d'attente éloignées qui possède la file d'attente cible AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

XMITQ (INQUOTE.XMIT.QUEUE)
Définit le nom de la file d'attente de transmission. Cette option est facultative ; si le nom d'une file d'attente de transmission n'est pas spécifié, une file d'attente portant le même nom que le gestionnaire de files d'attente éloignées est utilisée.

Dans les deux cas, la file d'attente de transmission appropriée doit être définie en tant que file d'attente locale avec un attribut *Usage* spécifiant qu'il s'agit d'une file d'attente de transmission (USAGE (XMITQ) dans les commandes MQSC).

Autre méthode d'insertion de messages dans une file d'attente éloignée

L'utilisation d'une définition locale d'une file d'attente éloignée n'est pas le seul moyen d'insérer des messages dans une file d'attente éloignée. Les applications peuvent spécifier le nom complet de la file d'attente, y compris le nom du gestionnaire de files d'attente éloignées, dans le cadre de l'appel à MQOPEN . Dans ce cas, vous n'avez pas besoin d'une définition locale d'une file d'attente éloignée. Toutefois, cela signifie que les applications doivent connaître le nom du gestionnaire de files d'attente éloignées ou y avoir accès lors de l'exécution.

Utilisation d'autres commandes avec des files d'attente distantes

Vous pouvez utiliser les commandes MQSC pour afficher ou modifier les attributs d'un objet de file d'attente éloignée, ou vous pouvez supprimer l'objet de file d'attente éloignée. Exemple :

- Pour afficher les attributs de la file d'attente éloignée:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- Pour modifier la file d'attente éloignée afin d'activer les insertions. Cela n'affecte pas la file d'attente cible, mais uniquement les applications qui spécifient cette file d'attente éloignée:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- Permet de supprimer cette file d'attente éloignée. Cela n'affecte pas la file d'attente cible, mais uniquement sa définition locale:

```
  DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Remarque : Lorsque vous supprimez une file d'attente éloignée, vous supprimez uniquement la représentation locale de la file d'attente éloignée. Vous ne supprimez pas la file d'attente éloignée elle-même ou les messages qu'elle contient.

Définition d'une file d'attente de transmission

Une file d'attente de transmission est une file d'attente locale utilisée lorsqu'un gestionnaire de files d'attente transfère des messages à un gestionnaire de files d'attente éloignées via un canal de transmission.

Le canal fournit une liaison unidirectionnelle vers le gestionnaire de files d'attente éloignées. Les messages sont placés dans la file d'attente de transmission jusqu'à ce que le canal puisse les accepter.

Lorsque vous définissez un canal, vous devez spécifier un nom de file d'attente de transmission à l'extrémité émettrice du canal de transmission.

L'attribut de commande MQSC USAGE définit si une file d'attente est une file d'attente de transmission ou une file d'attente normale.

Files d'attente de transmission par défaut

Lorsqu'un gestionnaire de files d'attente envoie des messages à un gestionnaire de files d'attente éloignées, il identifie la file d'attente de transmission dans l'ordre suivant :

1. La file d'attente de transmission nommée sur l'attribut XMITQ de la définition locale d'une file d'attente éloignée.
2. Une file d'attente de transmission portant le même nom que le gestionnaire de files d'attente cible. (Cette valeur est la valeur par défaut sur XMITQ de la définition locale d'une file d'attente éloignée.)
3. La file d'attente de transmission nommée sur l'attribut DEFXMITQ du gestionnaire de files d'attente local.

Par exemple, la commande MQSC suivante crée une file d'attente de transmission par défaut sur `source.queue.manager` pour les messages destinés à `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Les applications peuvent placer les messages directement dans une file d'attente de transmission ou indirectement via une définition de file d'attente éloignée. Voir aussi «Création d'une définition locale d'une file d'attente éloignée», à la page 116.

Utilisation de définitions de file d'attente éloignée comme alias

En plus de localiser une file d'attente sur un autre gestionnaire de files d'attente, vous pouvez également utiliser une définition locale d'une file d'attente éloignée pour les alias de gestionnaire de files d'attente et les alias de file d'attente de réponse. Les deux types d'alias sont résolus via la définition locale d'une file d'attente éloignée. Vous devez configurer les canaux appropriés pour que le message arrive à sa destination.

Alias de gestionnaire de files d'attente

Un alias est le processus par lequel le nom du gestionnaire de files d'attente cible, tel que spécifié dans un message, est modifié par un gestionnaire de files d'attente sur la route des messages.Les alias de gestionnaire de files d'attente sont importants car vous pouvez les utiliser pour contrôler la destination des messages au sein d'un réseau de gestionnaires de files d'attente.

Pour ce faire, modifiez la définition de file d'attente éloignée sur le gestionnaire de files d'attente au point de contrôle. L'application émettrice n'est pas consciente que le nom de gestionnaire de files d'attente spécifié est un alias.

Pour plus d'informations sur les alias de gestionnaire de files d'attente, voir Quels sont les alias?.

Alias de file d'attente de réponse

En option, une application peut spécifier le nom d'une file d'attente de réponse lorsqu'elle place un *message de demande* dans une file d'attente.

Si l'application qui traite le message extrait le nom de la file d'attente de réponse, elle sait où envoyer le *message de réponse*, si nécessaire.

Un alias de file d'attente de réponse est le processus par lequel une file d'attente de réponse, telle que spécifiée dans un message de demande, est modifiée par un gestionnaire de files d'attente sur la route

des messages. L'application émettrice ne sait pas que le nom de file d'attente de réponse indiqué est un alias.

Un alias de file d'attente de réponse vous permet de modifier le nom de la file d'attente de réponse et éventuellement de son gestionnaire de files d'attente. Cela vous permet de contrôler la route utilisée pour les messages de réponse.

Pour plus d'informations sur les messages de demande, les messages de réponse et les files d'attente de réponse, voir Types de message et File d'attente de réponse et gestionnaire de files d'attente.

Pour plus d'informations sur les alias de file d'attente de réponse, voir Alias de file d'attente de réponse et clusters .

Conversion de données entre des jeux de caractères codés

Les données de message dans les formats définis par WebSphere MQ (également appelés *formats intégrés*) peuvent être converties par le gestionnaire de files d'attente d'un jeu de caractères codés à un autre, à condition que les deux jeux de caractères soient associés à une langue unique ou à un groupe de langues similaires.

Par exemple, la conversion entre les jeux de caractères codés avec les identificateurs (CCSID) 850 et 500 est prise en charge, car les deux s'appliquent aux langues d'Europe occidentale.

Pour les conversions de caractères de nouvelle ligne (NL) EBCDIC en ASCII, voir Tous les gestionnaires de files d'attente .

Les conversions prises en charge sont définies dans Conversion de données.

Lorsqu'un gestionnaire de files d'attente ne parvient pas à convertir des messages dans des formats intégrés

Le gestionnaire de files d'attente ne peut pas convertir automatiquement les messages dans des formats intégrés si leurs CCSID représentent des groupes de langues nationales différents. Par exemple, la conversion entre le CCSID 850 et le CCSID 1025 (qui est un jeu de caractères codés EBCDIC pour les langues utilisant le script cyrillique) n'est pas prise en charge car la plupart des caractères d'un jeu de caractères codés ne peuvent pas être représentés dans l'autre. Si vous disposez d'un réseau de gestionnaires de files d'attente travaillant dans des langues nationales différentes et que la conversion de données entre certains jeux de caractères codés n'est pas prise en charge, vous pouvez activer une conversion par défaut. La conversion de données par défaut est décrite dans «Conversion de données par défaut», à la page 121.

Fichier ccsid.tbl

Le fichier ccsid.tbl est utilisé dans les buts suivants:

- Dans WebSphere MQ for Windows , il enregistre tous les jeux de codes pris en charge.
- Sur les plateformes AIX et HP-UX , les jeux de codes pris en charge sont conservés en interne par le système d'exploitation.
- Pour toutes les autres plateformes UNIX and Linux , les jeux de codes pris en charge sont conservés dans les tables de conversion fournies par WebSphere MQ.
- Il indique tous les jeux de codes supplémentaires. Pour spécifier des jeux de codes supplémentaires, vous devez éditer ccsid.tbl (des instructions sur la façon de procéder sont fournies dans le fichier).
- Il indique toute conversion de données par défaut.

Vous pouvez mettre à jour les informations enregistrées dans ccsid.tbl; vous pouvez le faire si, par exemple, une version ultérieure de votre système d'exploitation prend en charge des jeux de caractères codés supplémentaires.

Dans WebSphere MQ for Windows, ccsid.tbl se trouve par défaut dans le répertoire `C:\Program Files\IBM\WebSphere MQ\conv\table` .

Dans les systèmes WebSphere MQ for UNIX and Linux , ccsid.tbl se trouve dans le répertoire `/var/mqm/conv/table`.

Conversion de données par défaut

Si vous configurez des canaux entre deux machines sur lesquelles la conversion de données n'est pas normalement prise en charge, vous devez activer la conversion de données par défaut pour que les canaux fonctionnent.

Pour activer la conversion de données par défaut, éditez le fichier ccsid.tbl pour spécifier un CCSID EBCDIC par défaut et un CCSID ASCII par défaut. Des instructions sur la façon de procéder sont incluses dans le fichier. Vous devez effectuer cette opération sur toutes les machines qui seront connectées à l'aide des canaux. Redémarrez le gestionnaire de files d'attente pour que la modification soit prise en compte.

Le processus de conversion de données par défaut est le suivant:

- Si la conversion entre les CCSID source et cible n'est pas prise en charge, mais que les CCSID des environnements source et cible sont EBCDIC ou ASCII, les données de type caractères sont transmises à l'application cible sans conversion.
- Si un CCSID représente un jeu de caractères codés ASCII et que l'autre représente un jeu de caractères codés EBCDIC, WebSphere MQ convertit les données à l'aide des CCSID de conversion de données par défaut définis dans ccsid.tbl.

Remarque : Essayez de limiter les caractères convertis à ceux qui ont les mêmes valeurs de code dans le jeu de caractères codés indiqué pour le message et dans le jeu de caractères codés par défaut. Si vous utilisez uniquement l'ensemble de caractères qui est valide pour les noms d'objet WebSphere MQ (comme défini dans Nommage des objets IBM WebSphere MQ), vous respectez généralement cette exigence. Des exceptions se produisent avec les CCSID EBCDIC 290, 930, 1279 et 5026 utilisés au Japon, où les caractères minuscules ont des codes différents de ceux utilisés dans les autres CCSID EBCDIC.

Conversion de messages dans des formats définis par l'utilisateur

Le gestionnaire de files d'attente ne peut pas convertir des messages dans des formats définis par l'utilisateur d'un jeu de caractères codés à un autre. Si vous devez convertir des données dans un format défini par l'utilisateur, vous devez fournir un exit de conversion de données pour chaque format. N'utilisez pas les CCSID par défaut pour convertir les données de type caractères dans des formats définis par l'utilisateur. Pour plus d'informations sur la conversion de données dans des formats définis par l'utilisateur et sur l'écriture d'exits de conversion de données, voir Ecriture d'exits de conversion de données.

Modification du CCSID du gestionnaire de files d'attente

Une fois que vous avez utilisé l'attribut CCSID de la commande ALTER QMGR pour modifier le CCSID du gestionnaire de files d'attente, arrêtez et redémarrez le gestionnaire de files d'attente pour vous assurer que toutes les applications en cours d'exécution, y compris le serveur de commandes et les programmes de canal, sont arrêtées et redémarrées.

Cette opération est nécessaire car toutes les applications en cours d'exécution lorsque le CCSID du gestionnaire de files d'attente est modifié continuent d'utiliser le CCSID existant.

Administration de IBM WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry est administré à l'aide de IBM WebSphere MQ Explorer ou d'une ligne de commande. Utilisez l'explorateur pour configurer les canaux de télémétrie, contrôler le service de télémétrie et surveiller les clients MQTT qui sont connectés à IBM WebSphere MQ. Configurez la sécurité de IBM WebSphere MQ Telemetry à l'aide de JAAS, de SSL et du gestionnaire des droits d'accès aux objets IBM WebSphere MQ .

Administration à l'aide de IBM WebSphere MQ Explorer

Utilisez l'explorateur pour configurer les canaux de télémétrie, contrôler le service de télémétrie et surveiller les clients MQTT qui sont connectés à IBM WebSphere MQ. Configurez la sécurité de IBM WebSphere MQ Telemetry à l'aide de JAAS, de SSL et du gestionnaire des droits d'accès aux objets IBM WebSphere MQ .

Administration à l'aide de la ligne de commande

IBM WebSphere MQ Telemetry peut être entièrement administré à partir de la ligne de commande à l'aide des commandes IBM WebSphere MQ MQSC .

La documentation IBM WebSphere MQ Telemetry contient également des exemples de script qui illustrent l'utilisation de base de l'application client MQ Telemetry Transport v3 .

Lisez et comprenez les exemples dans les exemples de programmesIBM WebSphere MQ Telemetry de la section Développement d'applications pour IBM WebSphere MQ Telemetry avant de les utiliser.

Concepts associés

WebSphere MQ Telemetry

«Configuration des files d'attente réparties en vue de l'envoi de messages aux clients MQTT», à la page 126
Les applications IBM WebSphere MQ peuvent envoyer des messages aux clients MQTT v3 en les publiant dans un abonnement créé par un client ou en envoyant un message directement. Quelle que soit la méthode utilisée, le message est placé sur SYSTEM.MQTT.TRANSMIT.QUEUEet envoyé au client par le service de télémétrie (MQXR). Il existe plusieurs façons de placer un message sur SYSTEM.MQTT.TRANSMIT.QUEUE.

«Identification, autorisation et authentification du client MQTT», à la page 129

«Authentification du canal de télémétrie à l'aide de SSL», à la page 136

«Confidentialité de la publication sur les canaux de télémétrie», à la page 138

«Configuration SSL des clients MQTT et des canaux de télémétrie», à la page 139

«Configuration JAAS du canal de télémétrie», à la page 144
Configurez JAAS pour authentifier le `Nom d'utilisateur` envoyé par le client.

«Concepts du démon pour périphériques IBM WebSphere MQ Telemetry», à la page 146
Le démon pour dispositifs IBM WebSphere MQ Telemetry est une application client MQTT V3 avancée. Utilisez-le pour stocker et transférer des messages à d'autres clients MQTT. Il se connecte à IBM WebSphere MQ comme un client MQTT, mais vous pouvez également y connecter d'autres clients MQTT.

Tâches associées

«Configuration d'un gestionnaire de files d'attente pour la télémétrie sous Linux et AIX», à la page 122
Suivez ces étapes manuelles pour configurer un gestionnaire de files d'attente afin qu'il exécute IBM WebSphere MQ Telemetry. Vous pouvez exécuter une procédure automatisée pour définir une configuration plus simple à l'aide du support IBM WebSphere MQ Telemetry pour IBM WebSphere MQ Explorer.

«Configuration d'un gestionnaire de files d'attente pour la télémétrie sur Windows», à la page 124
Suivez ces étapes manuelles pour configurer un gestionnaire de files d'attente afin qu'il exécute IBM WebSphere MQ Telemetry. Vous pouvez exécuter une procédure automatisée pour définir une configuration plus simple à l'aide du support IBM WebSphere MQ Telemetry pour IBM WebSphere MQ Explorer.

Référence associée

Propriétés de MQXR

Configuration d'un gestionnaire de files d'attente pour la télémétrie sous Linux et AIX

Suivez ces étapes manuelles pour configurer un gestionnaire de files d'attente afin qu'il exécute IBM WebSphere MQ Telemetry. Vous pouvez exécuter une procédure automatisée pour définir une

configuration plus simple à l'aide du support IBM WebSphere MQ Telemetry pour IBM WebSphere MQ Explorer.

Avant de commencer

1. Pour plus d'informations sur l'installation de IBM WebSphere MQet de la fonction IBM WebSphere MQ Telemetry , voir Installation de IBM WebSphere MQ Telemetry .
2. Créez et démarrez un gestionnaire de files d'attente. Le gestionnaire de files d'attente est appelé *qMgr* dans cette tâche.
3. Dans le cadre de cette tâche, vous configurez le service de télémétrie (MQXR). Les paramètres de propriété MQXR sont stockés dans un fichier de propriétés spécifique à la plateforme: `mqxr_unix.properties`. Normalement, vous n'avez pas besoin d'éditer directement le fichier de propriétés MQXR, car presque tous les paramètres peuvent être configurés via des commandes d'administration MQSC ou MQ Explorer. Si vous décidez d'éditer le fichier directement, arrêtez le gestionnaire de files d'attente avant d'effectuer vos modifications. Voir MQXR properties.

Pourquoi et quand exécuter cette tâche

La IBM WebSphere MQ Telemetry prise en charge de IBM WebSphere MQ Explorer inclut un assistant et un exemple de procédure de commande `sampleMQM`. Ils définissent une configuration initiale à l'aide de l'ID utilisateur invité ; voir Vérification de l'installation de IBM WebSphere MQ Telemetry à l'aide des exemples de programme IBM WebSphere MQ Explorer et IBM WebSphere MQ Telemetry.

Suivez les étapes de cette tâche pour configurer IBM WebSphere MQ Telemetry manuellement à l'aide de différents schémas d'autorisation.

Procédure

1. Ouvrez une fenêtre de commande dans le répertoire des exemples de télémétrie.

   Le répertoire des exemples de télémétrie est `/opt/mqm/mqxr/samples`.
2. Créez la file d'attente de transmission de télémétrie.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc qMgr
   ```

   Lorsque le service de télémétrie (MQXR) est démarré pour la première fois, il crée `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Il est créé manuellement dans cette tâche, car `SYSTEM.MQTT.TRANSMIT.QUEUE` doit exister avant le démarrage du service de télémétrie (MQXR) pour autoriser l'accès à ce service.
3. Définir la file d'attente de transmission par défaut

   Lorsque le service de télémétrie (MQXR) est démarré pour la première fois, il ne modifie pas le gestionnaire de files d'attente pour faire de `SYSTEM.MQTT.TRANSMIT.QUEUE` la file d'attente de transmission par défaut.

   Pour que `SYSTEM.MQTT.TRANSMIT.QUEUE` soit la file d'attente de transmission par défaut, modifiez la propriété de file d'attente de transmission par défaut. Modifiez la propriété à l'aide de IBM WebSphere MQ Explorer ou de la commande de l'exemple suivant:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   La modification de la file d'attente de transmission par défaut peut interférer avec votre configuration existante. La modification de la file d'attente de transmission par défaut en `SYSTEM.MQTT.TRANSMIT.QUEUE` a pour but de faciliter l'envoi de messages directement aux clients MQTT. Sans modifier la file d'attente de transmission par défaut, vous devez ajouter une définition de file d'attente éloignée pour chaque client qui reçoit des messages IBM WebSphere MQ ; voir «Envoi direct d'un message à un client», à la page 128.

4. Suivez une procédure dans «Autorisation des clients MQTT à accéder aux objets WebSphere MQ», à la page 130 pour créer un ou plusieurs ID utilisateur. Les ID utilisateur sont autorisés à publier, à s'abonner et à envoyer des publications aux clients MQTT.
5. Installation du service de télémétrie (MQXR)

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   Voir aussi l'exemple de code dans Figure 19, à la page 124.
6. Démarrez le service

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   Le service de télémétrie (MQXR) est démarré automatiquement au démarrage du gestionnaire de files d'attente.

   Il est démarré manuellement dans cette tâche, car le gestionnaire de files d'attente est déjà en cours d'exécution.
7. A l'aide de IBM WebSphere MQ Explorer, configurez les canaux de télémétrie pour accepter les connexions des clients MQTT.

   Les canaux de télémétrie doivent être configurés de sorte que leurs identités soient l'un des ID utilisateur définis à l'étape 4.

   Voir aussi DEFINE CHANNEL (MQTT).
8. Vérifiez la configuration en exécutant l'exemple de client.

   Pour que l'exemple de client puisse utiliser votre canal de télémétrie, le canal doit autoriser le client à publier, s'abonner et recevoir des publications. L'exemple de client se connecte au canal de télémétrie sur le port 1883 par défaut. Voir aussi exemples de programmesIBM WebSphere MQ Telemetry.

Exemple

La Figure 19, à la page 124 illustre la commande **runmqsc** permettant de créer le SYSTEM.MQXR.SERVICE manuellement sous Linux.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
    STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+"') +
    STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
    STOPARG('-m +QMNAME+') +
    STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
    STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

Figure 19. installMQXRService_unix.mqsc

Configuration d'un gestionnaire de files d'attente pour la télémétrie sur Windows

Suivez ces étapes manuelles pour configurer un gestionnaire de files d'attente afin qu'il exécute IBM WebSphere MQ Telemetry. Vous pouvez exécuter une procédure automatisée pour définir une configuration plus simple à l'aide du support IBM WebSphere MQ Telemetry pour IBM WebSphere MQ Explorer.

Avant de commencer

1. Pour plus d'informations sur l'installation de IBM WebSphere MQet de la fonction IBM WebSphere MQ Telemetry , voir Installation de IBM WebSphere MQ Telemetry .
2. Créez et démarrez un gestionnaire de files d'attente. Le gestionnaire de files d'attente est appelé *qMgr* dans cette tâche.

3. Dans le cadre de cette tâche, vous configurez le service de télémétrie (MQXR). Les paramètres de propriété MQXR sont stockés dans un fichier de propriétés spécifique à la plateforme: `mqxr_win.properties`. Normalement, vous n'avez pas besoin d'éditer directement le fichier de propriétés MQXR, car presque tous les paramètres peuvent être configurés via des commandes d'administration MQSC ou MQ Explorer. Si vous décidez d'éditer le fichier directement, arrêtez le gestionnaire de files d'attente avant d'effectuer vos modifications. Voir MQXR properties.

Pourquoi et quand exécuter cette tâche

La IBM WebSphere MQ Telemetry prise en charge de IBM WebSphere MQ Explorer inclut un assistant et un exemple de procédure de commande `sampleMQM`. Ils définissent une configuration initiale à l'aide de l'ID utilisateur invité ; voir Vérification de l'installation de IBM WebSphere MQ Telemetry à l'aide des exemples de programme IBM WebSphere MQ Explorer et IBM WebSphere MQ Telemetry.

Suivez les étapes de cette tâche pour configurer IBM WebSphere MQ Telemetry manuellement à l'aide de différents schémas d'autorisation.

Procédure

1. Ouvrez une fenêtre de commande dans le répertoire des exemples de télémétrie.

   Le répertoire des exemples de télémétrie est *WMQ program installation directory*`\mqxr\samples`.
2. Créez la file d'attente de transmission de télémétrie.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   Lorsque le service de télémétrie (MQXR) est démarré pour la première fois, il crée `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Il est créé manuellement dans cette tâche, car `SYSTEM.MQTT.TRANSMIT.QUEUE` doit exister avant le démarrage du service de télémétrie (MQXR) pour autoriser l'accès à ce service.
3. Définir la file d'attente de transmission par défaut

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   Figure 20. Définir la file d'attente de transmission par défaut

   Lorsque le service de télémétrie (MQXR) est démarré pour la première fois, il ne modifie pas le gestionnaire de files d'attente pour faire de `SYSTEM.MQTT.TRANSMIT.QUEUE` la file d'attente de transmission par défaut.

   Pour que `SYSTEM.MQTT.TRANSMIT.QUEUE` soit la file d'attente de transmission par défaut, modifiez la propriété de file d'attente de transmission par défaut. Modifiez la propriété à l'aide de IBM WebSphere MQ Explorer ou de la commande dans Figure 20, à la page 125.

   La modification de la file d'attente de transmission par défaut peut interférer avec votre configuration existante. La modification de la file d'attente de transmission par défaut en `SYSTEM.MQTT.TRANSMIT.QUEUE` a pour but de faciliter l'envoi de messages directement aux clients MQTT. Sans modifier la file d'attente de transmission par défaut, vous devez ajouter une définition de file d'attente éloignée pour chaque client qui reçoit des messages IBM WebSphere MQ ; voir «Envoi direct d'un message à un client», à la page 128.
4. Suivez une procédure dans «Autorisation des clients MQTT à accéder aux objets WebSphere MQ», à la page 130 pour créer un ou plusieurs ID utilisateur. Les ID utilisateur sont autorisés à publier, à s'abonner et à envoyer des publications aux clients MQTT.
5. Installation du service de télémétrie (MQXR)

   ```
   type
   installMQXRService_win.mqsc | runmqsc qMgr
   ```
6. Démarrez le service

```
echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
```

Le service de télémétrie (MQXR) est démarré automatiquement au démarrage du gestionnaire de files d'attente.

Il est démarré manuellement dans cette tâche, car le gestionnaire de files d'attente est déjà en cours d'exécution.

7. A l'aide de IBM WebSphere MQ Explorer, configurez les canaux de télémétrie pour accepter les connexions des clients MQTT.

   Les canaux de télémétrie doivent être configurés de sorte que leurs identités soient l'un des ID utilisateur définis à l'étape 4.

   Voir aussi DEFINE CHANNEL (MQTT).
8. Vérifiez la configuration en exécutant l'exemple de client.

   Pour que l'exemple de client puisse utiliser votre canal de télémétrie, le canal doit autoriser le client à publier, s'abonner et recevoir des publications. L'exemple de client se connecte au canal de télémétrie sur le port 1883 par défaut. Voir aussi exemples de programmesIBM WebSphere MQ Telemetry.

Création manuelle de SYSTEM.MQXR.SERVICE

La Figure 21, à la page 126 présente la commande **runmqsc** permettant de créer le SYSTEM.MQXR.SERVICE manuellement sous Windows.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
    STARTARG('-m +QMNAME+  -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\."') +
    STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
    STOPARG('-m +QMNAME+') +
    STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
    STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

Figure 21. installMQXRService_win.mqsc

Configuration des files d'attente réparties en vue de l'envoi de messages aux clients MQTT

Les applications IBM WebSphere MQ peuvent envoyer des messages aux clients MQTT v3 en les publiant dans un abonnement créé par un client ou en envoyant un message directement. Quelle que soit la méthode utilisée, le message est placé sur SYSTEM.MQTT.TRANSMIT.QUEUEet envoyé au client par le service de télémétrie (MQXR). Il existe plusieurs façons de placer un message sur SYSTEM.MQTT.TRANSMIT.QUEUE.

Publication d'un message en réponse à un abonnement client MQTT

Le service de télémétrie (MQXR) crée un abonnement pour le compte du client MQTT. Le client est la destination des publications qui correspondent à l'abonnement envoyé par le client. Les services de télémétrie réachemine les publications correspondantes vers le client.

Un client MQTT est connecté à WebSphere MQ en tant que gestionnaire de files d'attente, son nom de gestionnaire de files d'attente étant défini sur ClientIdentifier. La destination des publications à envoyer au client est une file d'attente de transmission, SYSTEM.MQTT.TRANSMIT.QUEUE. Le service de télémétrie transmet les messages sur SYSTEM.MQTT.TRANSMIT.QUEUE aux clients MQTT, en utilisant le nom du gestionnaire de files d'attente cible comme clé d'un client spécifique.

Le service de télémétrie (MQXR) ouvre la file d'attente de transmission en utilisant ClientIdentifier comme nom de gestionnaire de files d'attente. Le service de télémétrie (MQXR) transmet le descripteur d'objet de la file d'attente à l'appel MQSUB pour transmettre les publications qui correspondent à l'abonnement du client. Dans la résolution de nom d'objet, ClientIdentifier est créé en tant que

nom de gestionnaire de files d'attente éloignées et la file d'attente de transmission doit être résolue en SYSTEM.MQTT.TRANSMIT.QUEUE. A l'aide de la résolution de nom d'objet WebSphere MQ standard, *ClientIdentifier* est résolu comme suit ; voir Tableau 6, à la page 127.

1. *ClientIdentifier* ne correspond à rien.

   ClientIdentifier est un nom de gestionnaire de files d'attente éloignées. Il ne correspond pas au nom du gestionnaire de files d'attente local, à un alias de gestionnaire de files d'attente ou à un nom de file d'attente de transmission.

   Le nom de la file d'attente n'est pas défini. Actuellement, le service de télémétrie (MQXR) définit SYSTEM.MQTT.PUBLICATION.QUEUE comme nom de la file d'attente. Un client MQTT v3 ne prenant pas en charge les files d'attente, le nom de la file d'attente résolue est ignoré par le client.

   La propriété du gestionnaire de files d'attente local, `File d'attente de transmission par défaut`, doit être définie sur SYSTEM.MQTT.TRANSMIT.QUEUE, de sorte que la publication soit placée sur SYSTEM.MQTT.TRANSMIT.QUEUE pour être envoyée au client.

2. *ClientIdentifier* correspond à un alias de gestionnaire de files d'attente nommé *ClientIdentifier*.

   ClientIdentifier est un nom de gestionnaire de files d'attente éloignées. Il correspond au nom d'un alias de gestionnaire de files d'attente.

   L'alias de gestionnaire de files d'attente doit être défini avec *ClientIdentifier* comme nom de gestionnaire de files d'attente éloignées.

   En définissant le nom de la file d'attente de transmission dans la définition d'alias de gestionnaire de files d'attente, il n'est pas nécessaire que la transmission par défaut soit définie sur SYSTEM.MQTT.TRANSMIT.QUEUE.

Tableau 6. Résolution de nom d'un alias de gestionnaire de files d'attente MQTT

	Entrée		**Sortie**		
ClientIdentifier	**Nom gest. de files**	**Nom de la file d'attente**	**Nom gest. de files**	**Nom de la file d'attente**	**File d'attente de transmission**
Ne correspond à rien	*ClientIdentifier*	*non défini*	*ClientIdentifier*	*non défini*	File d'attente de transmission par défaut. SYSTEM.MQTT.TRANSMIT.QUEUE
Correspond à un alias de gestionnaire de files d'attente nommé *ClientIdentifier*	*ClientIdentifier*	*non défini*	*ClientIdentifier*	*non défini*	SYSTEM.MQTT.TRANSMIT.QUEUE

Pour plus d'informations sur la résolution de nom, voir Résolution de nom.

Tout programme WebSphere MQ peut publier dans la même rubrique. La publication est envoyée à ses abonnés, y compris les clients MQTT v3 qui ont un abonnement à la rubrique.

Si une rubrique d'administration est créée dans un cluster, avec l'attribut CLUSTER(*clusterName*), toute application du cluster peut être publiée sur le client ; par exemple:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

Figure 22. Définition d'une rubrique de cluster sous Windows

Remarque : N'attribuez pas à SYSTEM.MQTT.TRANSMIT.QUEUE un attribut de cluster.

Les abonnés au client MQTT et les diffuseurs de publications peuvent se connecter à différents gestionnaires de files d'attente. Les abonnés et les diffuseurs peuvent faire partie du même cluster ou être connectés par une hiérarchie de publication / abonnement. La publication est fournie par le diffuseur de publications à l'abonné à l'aide de WebSphere MQ.

Envoi direct d'un message à un client

Une alternative à un client qui crée un abonnement et reçoit une publication qui correspond à la rubrique d'abonnement, envoie directement un message à un client MQTT v3 . Les applications client MQTT V3 ne peuvent pas envoyer de messages directement, contrairement à d'autres applications, telles que WebSphere MQ .

L'application WebSphere MQ doit connaître l'identificateur `ClientIdentifier` du client MQTT v3 . Comme les clients MQTT v3 n'ont pas de files d'attente, le nom de la file d'attente cible est transmis à la méthode MQTT v3 client d'application `messageArrived` en tant que nom de rubrique. Par exemple, dans un programme MQI, créez un descripteur d'objet avec le client comme `ObjectQmgrNom`:

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

Figure 23. Descripteur d'objet MQI permettant d'envoyer un message à une destination de client MQTT v3

Si l'application est écrite à l'aide de JMS, créez une destination point-à-point, par exemple:

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

Figure 24. Destination JMS pour l'envoi d'un message à un client MQTT v3

Pour envoyer un message non sollicité à un client MQTT, utilisez une définition de file d'attente éloignée. Le nom du gestionnaire de files d'attente éloignées doit être résolu en `ClientIdentifier` du client. La file d'attente de transmission doit être résolue en SYSTEM.MQTT.TRANSMIT.QUEUE; voir Tableau 7, à la page 128. Le nom de la file d'attente éloignée peut être n'importe quel nom. Le client le reçoit sous forme de chaîne de rubrique.

Tableau 7. Résolution de nom d'une définition de file d'attente éloignée de client MQTT

Entrée		Sortie		
Nom de la file d'attente	**Nom gest. de files**	**Nom de la file d'attente**	**Nom gest. de files**	**File d'attente de transmission**
Nom de la définition de file d'attente éloignée	Nom de gestionnaire de files d'attente local ou vide	Nom de file d'attente éloignée utilisé comme chaîne de rubrique	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

Si le client est connecté, le message est envoyé directement au client MQTT, qui appelle la méthode `messageArrived` ; voir MéthodemessageArrived.

Si le client s'est déconnecté avec une session persistante, le message est stocké dans SYSTEM.MQTT.TRANSMIT.QUEUE; voir Sessions avec état et sans état MQTT . Il est transmis au client lorsque ce dernier se reconnecte à nouveau à la session.

Si vous envoyez un message non persistant, il est envoyé au client avec la qualité de service "au plus une fois", QoS=0. Si vous envoyez un message persistant directement à un client, par défaut, il est envoyé avec la qualité de service "une seule fois", QoS=2. Comme le client n'a peut-être pas de mécanisme de persistance, il peut diminuer la qualité de service qu'il accepte pour les messages envoyés directement. Pour réduire la qualité de service des messages envoyés directement à un client, abonnez-vous à la rubrique DEFAULT.QoS. Indiquez la qualité de service maximale que le client peut prendre en charge.

Identification, autorisation et authentification du client MQTT

Le service de télémétrie (MQXR) diffuse des publications ou s'abonne aux rubriques WebSphere MQ au nom des clients MQTT, à l'aide des canaux MQTT. L'administrateur WebSphere MQ configure l'identité du canal MQTT qui est utilisé pour l'autorisation WebSphere MQ. L'administrateur peut définir une identité commune pour le canal ou utiliser le Nom d'utilisateur ou l'Identificateur client d'un client connecté au canal.

Le service de télémétrie (MQXR) peut authentifier le client à l'aide du Nom d'utilisateur fourni par le client ou à l'aide d'un certificat client. Le Nom d'utilisateur est authentifié à l'aide d'un mot de passe fourni par le client.

En résumé : l'identification client est la sélection de l'identité du client. En fonction du contexte, le client est identifié par l'Identificateur client, le Nom d'utilisateur, une identité client commune créée par l'administrateur ou un certificat client. L'identificateur client utilisé pour le contrôle d'authenticité ne doit pas être le même que celui utilisé pour l'autorisation.

Les programmes du client MQTT définissent les Nom d'utilisateur et Mot de passe qui sont envoyés au serveur à l'aide d'un canal MQTT. Ils peuvent également définir les propriétés SSL requises pour chiffrer et authentifier la connexion. L'administrateur décide s'il est nécessaire d'authentifier le canal MQTT et comment procéder.

Pour autoriser un client MQTT à accéder aux objets WebSphere MQ, accordez l'autorisation d'accès à l'Identificateur client ou au nom d'utilisateur, ou accordez l'autorisation à une identité client commune. Pour autoriser un client à se connecter à WebSphere MQ, vous devez authentifier le nom d'utilisateur ou utiliser un certificat client. Configurez JAAS pour authentifier le Nom d'utilisateur et configurer SSL pour authentifier un certificat client.

Si vous définissez un Mot de passe au niveau du client, vous devez chiffrer la connexion à l'aide du réseau privé virtuel (VPN) ou configurer le canal MQTT pour utiliser SSL, afin que le mot de passe reste privé.

Il est difficile de gérer des certificats client. Pour cette raison, si les risques associés à l'authentification par mot de passe sont acceptables, ce type d'authentification est souvent utilisé pour l'authentification des clients.

S'il existe un moyen sûr de gérer et de stocker le certificat client, il est possible de faire confiance à l'authentification par certificat. Toutefois, il arrive rarement que les certificats puissent être gérés de manière sécurisée dans les types d'environnements dans lesquels la télémétrie est utilisée. L'authentification des dispositifs utilisant les certificats client est complétée par l'authentification des mots de passe client au niveau du serveur. En raison de la complexité supplémentaire, l'utilisation du certificat client est limitée aux applications hautement sensibles. L'utilisation de deux formulaires d'authentification est appelée authentification à deux facteurs. Vous devez connaître l'un des facteurs, tels que le mot de passe, et disposer de l'autre, un certificat, par exemple.

Dans une application sensible, telle qu'un appareil utilisant un code confidentiel, l'appareil est verrouillé pendant sa fabrication pour empêcher toute contrefaçon avec le matériel et le logiciel internes. Un certificat client digne de confiance, limité dans le temps, est copié dans le dispositif. Le dispositif est déployé à l'emplacement où il doit être utilisé. Une authentification supplémentaire est effectuée chaque fois que le dispositif est utilisé, soit à l'aide d'un mot de passe, soit en utilisant un autre certificat à partir d'une carte à puce.

MQTT client - Identité et autorisation

Utilisez l'`Identificateur client`, le `Nom d'utilisateur` ou une identité client commune pour accorder l'autorisation d'accès aux objets WebSphere MQ.

L'administrateur WebSphere MQ dispose de trois options pour sélectionner l'identité du canal MQTT. L'administrateur effectue son choix lors de la définition ou de la modification du canal MQTT utilisé par le client. L'identité est utilisée pour autoriser l'accès aux rubriques WebSphere MQ. Les choix possibles sont les suivants :

1. L'identificateur client.
2. Une identité fournie au canal par l'administrateur.
3. Le `Nom d'utilisateur` transmis à partir du client MQTT.

Le `Nom d'utilisateur` est un attribut de la classe MqttConnectOptions. Il doit être défini avant que le client ne se connecte au service. Sa valeur par défaut est Null.

Utilisez la commande WebSphere MQ **setmqaut** pour sélectionner les objets et les actions qui ont l'autorisation d'être utilisés par l'identité associée au canal MQTT. Par exemple, pour autoriser une identité de canal, `MQTTClient`, fournie par l'administrateur de gestionnaire de files d'attente, `QM1`:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

Autorisation des clients MQTT à accéder aux objets WebSphere MQ

Procédez comme suit pour autoriser les clients MQTT à publier des objets WebSphere MQ et à s'y abonner. Ces étapes décrivent quatre modèles de contrôle d'accès différents.

Avant de commencer

Les clients MQTT sont autorisés à accéder aux objets dans WebSphere MQ en se faisant affecter une identité lorsqu'ils se connectent à un canal de télémétrie. L'administrateur WebSphere MQ configure le canal de télémétrie à l'aide de WebSphere MQ Explorer pour donner à un client l'un des trois types d'identité suivants:

1. `ClientIdentifier`
2. `Nom d'utilisateur`
3. Nom que l'administrateur affecte au canal.

Quel que soit le type utilisé, l'identité doit être définie pour WebSphere MQ en tant que principal par le service d'autorisation installé. Le service d'autorisation par défaut sous Windows ou Linux est appelé Object Authority Manager (OAM). Si vous utilisez la méthode d'accès aux objets (OAM), l'identité doit être définie en tant qu'ID utilisateur.

Utilisez l'identité pour accorder à un client ou à une collection de clients le droit de publier des rubriques définies dans WebSphere MQou de s'y abonner. Si un client MQTT s'est abonné à une rubrique, utilisez l'identité pour lui accorder le droit de recevoir les publications résultantes.

Il est difficile de gérer un système avec des dizaines de milliers de clients MQTT, chacun nécessitant des droits d'accès individuels. Une solution consiste à définir des identités communes et à associer des clients MQTT individuels à l'une des identités communes. Définissez autant d'identités communes que nécessaire pour définir différentes combinaisons de droits. Une autre solution consiste à écrire votre propre service d'autorisation qui peut traiter plus facilement avec des milliers d'utilisateurs que le système d'exploitation.

Vous pouvez combiner des clients MQTT en identités communes de deux manières, à l'aide de la méthode d'accès aux objets (OAM):

1. Définissez plusieurs canaux de télémétrie, chacun avec un ID utilisateur différent que l'administrateur alloue à l'aide de WebSphere MQ Explorer. Les clients qui se connectent à l'aide de numéros de port TCP/IP différents sont associés à des canaux de télémétrie différents et se voient attribuer des identités différentes.

2. Définissez un canal de télémétrie unique, mais chaque client doit sélectionner un `nom d'utilisateur` dans un petit ensemble d'ID utilisateur. L'administrateur configure le canal de télémétrie pour sélectionner le `nom d'utilisateur` du client comme identité.

Dans cette tâche, l'identité du canal de télémétrie est appelée *mqttUser*, quel que soit le mode de définition. Si les collections de clients utilisent des identités différentes, utilisez plusieurs *mqttUsers*, un pour chaque collection de clients. Comme la tâche utilise la méthode d'accès aux objets (OAM), chaque *mqttUser* doit être un ID utilisateur.

Pourquoi et quand exécuter cette tâche

Dans cette tâche, vous avez le choix entre quatre modèles de contrôle d'accès que vous pouvez adapter à des exigences spécifiques. Les modèles diffèrent dans leur granularité du contrôle d'accès.

- «Aucun contrôle d'accès», à la page 131
- «Contrôle d'accès à granularité grossière», à la page 131
- «Contrôle d'accès à granularité moyenne», à la page 131
- «Contrôle d'accès à granularité fine», à la page 132

Le résultat des modèles est d'affecter des *mqttUsers* ensembles de droits de publication et d'abonnement à WebSphere MQet de recevoir des publications de WebSphere MQ.

Aucun contrôle d'accès
Les clients MQTT disposent des droits d'administration WebSphere MQ et peuvent effectuer n'importe quelle action sur n'importe quel objet.

Procédure

1. Créez un ID utilisateur *mqttUser* pour agir en tant qu'identité de tous les clients MQTT.
2. Ajoutez *mqttUser* au groupe `mqm` ; voir Ajout d'un utilisateur à un groupe sous Windowsou Ajout d'un utilisateur à un groupe sous Linux

Contrôle d'accès à granularité grossière
Les clients MQTT sont autorisés à publier et à s'abonner et à envoyer des messages aux clients MQTT. Ils n'ont pas le droit d'effectuer d'autres actions ou d'accéder à d'autres objets.

Procédure

1. Créez un ID utilisateur *mqttUser* pour agir en tant qu'identité de tous les clients MQTT.
2. Autorisez *mqttUser* à publier et à s'abonner à toutes les rubriques et à envoyer des publications aux clients MQTT.

```
setmqaut -m qMgr -t topic -n SYSTEM.BASE.TOPIC -p mqttUser -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqttUser -all +put
```

Contrôle d'accès à granularité moyenne
Les clients MQTT sont divisés en différents groupes pour publier et s'abonner à différents ensembles de rubriques, et pour envoyer des messages aux clients MQTT.

Procédure

1. Créez plusieurs ID utilisateur, *mqttUsers*et plusieurs rubriques d'administration dans l'arborescence de rubriques de publication / abonnement.
2. Autoriser différents *mqttUsers* à accéder à des rubriques différentes.

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. Créez un groupe *mqtt*et ajoutez-y tous les *mqttUsers* .

4. Autorisez *mqtt* à envoyer des rubriques aux clients MQTT.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Contrôle d'accès à granularité fine
Les clients MQTT sont incorporés dans un système de contrôle d'accès existant, qui autorise les groupes à effectuer des actions sur les objets.

Pourquoi et quand exécuter cette tâche

Un ID utilisateur est affecté à un ou plusieurs groupes de système d'exploitation en fonction des autorisations requises. Si les applications WebSphere MQ publient et s'abonnent au même espace de sujet que les clients MQTT, utilisez ce modèle. Les groupes sont appelés Publish*X*, Subscribe*Y*et `mqtt`

Publish*X*
Les membres des groupes Publish*X* peuvent publier dans *topicX*.

Subscribe*Y*
Les membres des groupes Subscribe*Y* peuvent s'abonner à *topicY*.

mqtt
Les membres du groupe *mqtt* peuvent envoyer des publications aux clients MQTT.

Procédure

1. Créez plusieurs groupes, Publish*X* et Subscribe*Y* , qui sont alloués à plusieurs rubriques d'administration dans l'arborescence de rubriques de publication / abonnement.
2. Créez un groupe `mqtt`.
3. Créez plusieurs ID utilisateur, *mqttUsers*, et ajoutez les utilisateurs à n'importe quel groupe, en fonction de ce qu'ils sont autorisés à faire.
4. Autorisez différents groupes Publish*X* et Subscribe*X* à accéder à des rubriques différentes et autorisez le groupe *mqtt* à envoyer des messages aux clients MQTT.

```
setmqaut -m qMgr -t topic -n topic1 -p PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Authentification du client MQTT à l'aide d'un mot de passe

Authentifiez le `Nom d'utilisateur` à l'aide du mot de passe du client. Vous pouvez authentifier le client en utilisant une identité différente de celle utilisée pour autoriser le client à diffuser des publications et à s'abonner aux sujets.

Le service de télémétrie (MQXR) utilise JAAS pour authentifier le `Nom d'utilisateur` du client. JAAS utilise le `mot de passe` fourni par le client MQTT.

L'administrateur WebSphere MQ décide s'il convient d'authentifier le `Nom d'utilisateur`, ou de ne pas l'authentifier du tout en configurant le canal MQTT auquel le client se connecte. Les clients peuvent se voir attribuer des canaux différents et chaque canal peut être configuré pour authentifier ses clients de différentes façons. A l'aide de JAAS, vous pouvez décider quelles sont les méthodes qui doivent authentifier le client, et celles qui peuvent authentifier le client.

Le choix de l'identité pour l'authentification n'affecte pas le choix de l'identité pour l'autorisation. Vous pouvez définir une identité commune pour l'autorisation à des fins d'administration, mais authentifier chaque utilisateur afin qu'il puisse utiliser cette identité. La procédure suivante met en évidence les étapes permettant d'authentifier des utilisateurs individuels pour l'utilisation d'une identité commune :

1. L'administrateur WebSphere MQ définit une identité de canal MQTT pour chaque nom, tel que `MQTTClientUser`, à l'aide de WebSphere MQ Explorer.

2. L'administrateur WebSphere MQ autorise `MQTTClient` à diffuser des publications et à s'abonner à n'importe quelle rubrique :

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

3. Le développeur d'applications client MQTT crée un objet `MqttConnectOptions` et définit `Username` et `Password` avant de se connecter au serveur.
4. Le développeur de sécurité crée un JAAS `LoginModule` pour authentifier le `nom d'utilisateur` avec le `mot de passe` et l'inclut dans le fichier de configuration JAAS .
5. L'administrateur WebSphere MQ configure le canal MQTT pour authentifier le `Nom d'utilisateur` du client à l'aide de JAAS.

Authentification du client MQTT à l'aide de SSL

Les connexions entre le client MQTT et le gestionnaire de files d'attente sont toujours initiées par le client MQTT. Le client MQTT est toujours le client SSL. L'authentification de client du serveur et l'authentification de serveur du client MQTT sont toutes les deux facultatives.

En fournissant au client un certificat numérique signé privé, vous pouvez authentifier le client MQTT auprès d' IBM WebSphere MQ. L'administrateur d'IBM WebSphere MQ peut forcer les clients MQTT à s'authentifier sur le gestionnaire de files d'attente à l'aide de SSL. Vous pouvez uniquement demander l'authentification de client comme faisant partie d'une authentification mutuelle.

Comme alternative à l'utilisation de SSL, certains types de réseaux privés virtuels (VPN), tels que IPsec, authentifient les noeuds finaux d'une connexion TCP/IP. Le réseau privé virtuel VPN chiffre chaque paquet transitant sur le réseau. Une fois qu'une telle connexion VPN est établie, vous avez établi un réseau sécurisé. Vous pouvez connecter les clients MQTT aux canaux de télémétrie avec TCP/IP sur le réseau VPN.

L'authentification de client à l'aide de SSL suppose que le client possède une clé confidentielle. La clé confidentielle est la clé privée du client dans le cas d'un certificat autosigné ou une clé fournie par une autorité de certification. La clé permet de signer le certificat numérique du client. Toute personne en possession de la clé publique correspondante peut vérifier le certificat numérique. Les certificats peuvent être accrédités, ou s'ils font partie d'une chaîne, il est possible de remonter par l'intermédiaire de la chaîne jusqu'au certificat racine accrédité. L'opération de vérification du client envoie tous les certificats de la chaîne fournie par le client au serveur. Le serveur vérifie la chaîne de certificats jusqu'à ce qu'il trouve un certificat digne de confiance. Il s'agit soit du certificat public généré par un certificat autosigné, soit d'un certificat racine généralement émis par une autorité de certification. Au cours de l'étape finale facultative le certificat de confiance peut être comparé à une liste de révocation de certificats "en direct".

Le certificat de confiance peut être émis par une autorité de certification et inclus dans l'espace de stockage de certificats de JRE. Il peut s'agir d'un certificat autosigné ou de tout certificat qui a été ajouté au magasin de clés de canal de télémétrie en tant que certificat digne de confiance.

Remarque : Le canal de télémétrie dispose d'un magasin servant à la fois de magasin de clés et de magasin de clés de confiance, qui contient les clés privées d'accès à un ou plusieurs canaux de télémétrie et les certificats publics nécessaires pour authentifier les clients. Comme un canal SSL doit être associé à un magasin de clés et que ce magasin est le même fichier que le magasin de clés de confiance du canal, l'espace de stockage des certificats du JRE n'est jamais référencé. Il en résulte que si l'authentification d'un client requiert un certificat racine de l'autorité de certification, vous devez placer ce certificat dans le magasin de clés du canal, même si le certificat racine de l'autorité de certification est déjà présent dans l'espace de stockage des certificats du JRE. L'espace de stockage des certificats du JRE n'est jamais référencé.

Ayez à l'esprit les menaces que l'authentification de client peut déjouer et le rôle du client et du serveur pour faire échouer ces menaces. L'authentification du certificat client en soi ne suffit pas à empêcher l'accès non autorisé à un système. Si quelqu'un d'autre a pris le possession du dispositif client, ce dernier ne s'exécute pas nécessairement avec les droits du détenteur du certificat. Ne faites jamais confiance à une protection unique contre des attaques non désirées. Utilisez au moins une approche de type authentification à deux facteurs et ajoutez en supplément à la possession d'un certificat la connaissance

d'informations privées. Par exemple, utilisez JAAS, et authentifiez le client à l'aide d'un mot de passe émis par le serveur.

La principale menace à laquelle est confronté le certificat client est de se trouver aux mains de quelqu'un de mal intentionné. Le certificat se trouve dans un magasin de clés protégé par mot de passe au niveau du client. Comment se retrouve-t-il dans le magasin de clés ? Comment le client MQTT parvient-il à se procurer le mot de passe du magasin de clés ? Quel est le degré de sécurité de la protection par mot de passe ? Les dispositifs de télémétrie sont faciles à supprimer et peuvent donc être piratés en privé. Le dispositif matériel peut-il être imperméable aux actes de malveillance ? La distribution et la protection de certificat côté client sont reconnues comme un acte difficile, on parle alors de problème gestion de clés.

La menace secondaire est le cas ou le dispositif est mal utilisé pour accéder au serveur de manière non intentionnelle. Par exemple, si l'application MQTT est trafiquée il est possible d'utiliser une faiblesse dans la configuration du serveur à l'aide de l'identité du client authentifié.

Pour authentifier un client MQTT à l'aide de SSL, configurez le canal de télémétrie et le client.

•
•

Configuration du canal de télémétrie pour l'authentification du client MQTT par SSL

L'administrateur IBM WebSphere MQ configure les canaux de télémétrie sur le serveur. Chaque canal est configuré de manière à accepter une connexion TCP/IP sur un numéro de port différent. Les canaux SSL sont configurés avec accès au fichier de clés protégé par phrase passe. Si un canal SSL est défini sans phrase passe ou fichier de clés, le canal n'accepte pas de connexion SSL.

Définissez la propriété `com.ibm.mq.MQTT.ClientAuth` d'un canal de télémétrie SSL sur `REQUIRED` pour forcer tous les clients à se connecter sur ce canal afin de fournir la preuve qu'ils ont vérifié des certificats numériques. Les certificats client sont authentifiés à l'aide des certificats des autorités de certification, ce qui permet d'obtenir un certificat racine digne de confiance. Si le certificat client est autosigné ou signé par un certificat provenant d'une autorité de certification, les certificats signés publiquement du client ou de l'autorité de certification doivent être stockés de manière sécurisée sur le serveur.

Placez le certificat client signé publiquement ou le certificat de l'autorité de certification dans le magasin de clés du canal de télémétrie. Sur le serveur, les certificats signés publiquement sont stockés dans le même fichier de clés que les certificats signés en privé, plutôt que dans un magasin de clés de confiance distinct.

Le serveur vérifie la signature des certificats client qu'il envoie à l'aide de tous les certificats publics et des suites de chiffrement dont il dispose. Le serveur vérifie la chaîne de clés. Le gestionnaire de files d'attente peut être configuré pour tester le certificat par rapport à la liste de révocation de certificat. La propriété de liste de noms de révocation du gestionnaire de files d'attente est `SSLCRLNL`.

Si l'un des certificats envoyés par un client est vérifié par un certificat dans le magasin de clés du serveur, le client est authentifié.

L'administrateur WebSphere MQ peut configurer le même canal de télémétrie pour utiliser JAAS afin de vérifier le `UserName` ou `ClientIdentifier` du client avec le `Mot de passe`du client.

Vous pouvez utiliser le même magasin de clés pour plusieurs canaux de télémétrie.

La vérification d'au moins un certificat numérique dans le magasin de clés du client protégé par mot de passe sur le périphérique authentifie le client auprès du serveur. Le certificat numérique est utilisé uniquement pour l'authentification par WebSphere MQ. Il n'est pas utilisé pour vérifier l'adresse TCP/IP du client, ni pour définir l'identité du client pour l'autorisation ou la comptabilité. L'identité du client adoptée par le serveur est `Username` ou `ClientIdentifier` du client, ou une identité créée par l'administrateur WebSphere MQ .

Vous pouvez également utiliser les suites de chiffrement SSL pour l'authentification du client.Voici la liste alphabétique des suites de chiffrement actuellement prises en charge :

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`

- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V 7.5.0.2 Si vous prévoyez d'utiliser les algorithmes de cryptographie SHA-2 , voir Configuration système requise pour l'utilisation des algorithmes de chiffrement SHA-2 avec les canaux MQTT.

Concepts associés

«Configuration du canal de télémétrie pour l'authentification du canal par SSL», à la page 136
L'administrateur IBM WebSphere MQ configure les canaux de télémétrie sur le serveur. Chaque canal est configuré de manière à accepter une connexion TCP/IP sur un numéro de port différent. Les canaux SSL sont configurés avec accès au fichier de clés protégé par phrase passe. Si un canal SSL est défini sans phrase passe ou fichier de clés, le canal n'accepte pas de connexion SSL.

CipherSpecs et CipherSuites

Référence associée

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

Authentification du canal de télémétrie à l'aide de SSL

Les connexions entre le client MQTT et le gestionnaire de files d'attente sont toujours initiées par le client MQTT. Le client MQTT est toujours le client SSL. L'authentification de client du serveur et l'authentification de serveur du client MQTT sont toutes les deux facultatives.

Le client tente toujours d'authentifier le serveur, sauf si le client est configuré pour utiliser un CipherSpec qui prend en charge la connexion anonyme. Si l'authentification échoue, la connexion n'est pas établie.

Comme alternative à l'utilisation de SSL, certains types de réseaux privés virtuels (VPN), tels que IPsec, authentifient les noeuds finaux d'une connexion TCP/IP. Le réseau privé virtuel VPN chiffre chaque paquet transitant sur le réseau. Une fois qu'une telle connexion VPN est établie, vous avez établi un réseau sécurisé. Vous pouvez connecter les clients MQTT aux canaux de télémétrie avec TCP/IP sur le réseau VPN.

L'authentification de serveur à l'aide de SSL permet d'authentifier le serveur auquel vous êtes sur le point d'envoyer des informations confidentielles. Le client effectue les vérifications correspondant aux certificats envoyés à partir du serveur, par rapport aux certificats placés dans son magasin de clés de confiance ou dans son magasin JRE `cacerts` .

Le magasin de certificats JRE est un fichier JKS, `cacerts`. Il se trouve dans `JRE InstallPath\lib\security\`. Il est installé avec le mot de passe par défaut `changeit`. Vous pouvez stocker les certificats que vous accréditez dans l'espace de stockage des certificats JRE ou dans le magasin de clés de confiance. Vous ne pouvez pas utiliser les deux magasins. Utilisez le fichier de clés certifiées si vous souhaitez conserver les certificats publics et les certificats utilisés par d'autres applications Java à des emplacements distincts. Utilisez l'espace de stockage des certificats JRE si vous souhaitez utiliser un certificat commun pour toutes les applications Java qui s'exécutent sur le client. Si vous décidez d'utiliser cet espace de stockage, vérifiez les certificats qu'il contient afin d'être sûr que vous les accréditez.

Vous pouvez modifier la configuration JSSE en indiquant un autre fournisseur d'accréditation. Vous pouvez personnaliser un fournisseur d'accréditation pour effectuer différentes vérifications sur un certificat. Dans certains environnements OGSi qui ont utilisé le client MQTT, l'environnement fournit un fournisseur d'accréditation différent.

Pour authentifier un canal de télémétrie à l'aide de SSL, configurez le serveur et le client.

-
-

Configuration du canal de télémétrie pour l'authentification du canal par SSL

L'administrateur IBM WebSphere MQ configure les canaux de télémétrie sur le serveur. Chaque canal est configuré de manière à accepter une connexion TCP/IP sur un numéro de port différent. Les canaux SSL sont configurés avec accès au fichier de clés protégé par phrase passe. Si un canal SSL est défini sans phrase passe ou fichier de clés, le canal n'accepte pas de connexion SSL.

Stockez le certificat numérique du serveur, signé avec sa clé privée, dans le magasin de clés que le canal de télémétrie va utiliser sur le serveur. Stockez tous les certificats de sa chaîne de clés dans le magasin de clés, si vous souhaitez transmettre la chaîne de clés au client. Configurez le canal de télémétrie à l'aide de l'explorateur WebSphere MQ pour utiliser SSL. Indiquez le chemin d'accès au magasin de clés et la phrase passe pour accéder au magasin de clés. Si vous ne définissez pas le numéro de port TCP/IP du canal, le numéro de port du canal de télémétrie SSL est par défaut 8883.

Vous pouvez également utiliser les suites de chiffrement SSL pour l'authentification du canal.Voici la liste alphabétique des suites de chiffrement actuellement prises en charge :

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
-  SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
-  SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
-  SSL_RSA_WITH_AES_128_CBC_SHA256
-  SSL_RSA_WITH_AES_256_CBC_SHA256

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 Si vous prévoyez d'utiliser les algorithmes de cryptographie SHA-2 , voir Configuration système requise pour l'utilisation des algorithmes de chiffrement SHA-2 avec les canaux MQTT.

Concepts associés

«Configuration du canal de télémétrie pour l'authentification du client MQTT par SSL», à la page 134
L'administrateur IBM WebSphere MQ configure les canaux de télémétrie sur le serveur. Chaque canal est configuré de manière à accepter une connexion TCP/IP sur un numéro de port différent. Les canaux SSL sont configurés avec accès au fichier de clés protégé par phrase passe. Si un canal SSL est défini sans phrase passe ou fichier de clés, le canal n'accepte pas de connexion SSL.

CipherSpecs et CipherSuites

Référence associée

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)

Confidentialité de la publication sur les canaux de télémétrie

La confidentialité des publications MQTT transitant dans les canaux de télémétrie est assurée par l'utilisation de SSL pour le chiffrement des transmissions au cours de la connexion.

Les clients MQTT qui se connectent aux canaux de télémétrie utilisent SSL pour assurer la sécurité de la confidentialité des publications transmises sur le canal à l'aide du chiffrement par clé symétrique. Etant donné que les noeuds finaux ne sont pas authentifiés, vous ne pouvez pas accréditer le chiffrement de canal seul. Combinez la sécurisation de la confidentialité avec l'authentification du serveur ou l'authentification mutuelle.

Comme alternative à l'utilisation de SSL, certains types de réseaux privés virtuels (VPN), tels que IPsec, authentifient les noeuds finaux d'une connexion TCP/IP. Le réseau privé virtuel VPN chiffre chaque paquet transitant sur le réseau. Une fois qu'une telle connexion VPN est établie, vous avez établi un réseau sécurisé. Vous pouvez connecter les clients MQTT aux canaux de télémétrie avec TCP/IP sur le réseau VPN.

Pour une configuration classique, qui chiffre le canal et authentifie le serveur, consultez la section «Authentification du canal de télémétrie à l'aide de SSL», à la page 136.

Le chiffrement de connexions SSL sans authentification du serveur expose la connexion à des attaques de type "man in the middle". Bien que les informations que vous échangez soient protégées contre les écoutes clandestines, vous ne savez pas avec qui vous réalisez les échanges. A moins que vous ne contrôliez le réseau, vous êtes exposé à ce que quelqu'un intercepte vos transmissions IP, en se faisant passer pour un noeud final.

Vous pouvez créer une connexion SSL cryptée sans authentifier le serveur, à l'aide d'une spécification de chiffrement d'échange de clé Diffie-Hellman qui prend en charge SSL anonyme. Le secret maître, partagé entre le client et le serveur et utilisé pour chiffrer les transmissions SSL, est établi sans échange de certificat de serveur signé de manière privée.

Etant donné que mes connexions anonymes ne sont pas sécurisées, la plupart des implémentations SSL n'utilisent pas par défaut les spécifications CipherSpecs anonymes. Si une demande client de connexion SSL est acceptée par le canal de télémétrie, ce dernier doit disposer d'un magasin de clés protégé par une phrase passe. Par défaut, étant donné que les implémentations n'utilisent pas de spécification de chiffrement anonyme, le magasin de clés doit contenir un certificat signé de manière privée que le client peut authentifier.

Si vous utilisez des spécifications CipherSpecs anonymes, le magasin de clés du serveur doit exister, mais il ne doit pas forcément contenir des certificats signés de manière privée.

Une autre façon d'établir une connexion cryptée consiste à remplacer le fournisseur d'accréditation côté client par votre propre implémentation. Votre fournisseur d'accréditation n'effectuerait pas l'authentification du certificat serveur, mais la connexion serait cryptée.

Configuration SSL des clients MQTT et des canaux de télémétrie

Les clients MQTT et le service WebSphere MQ Telemetry (MQXR) utilisent JSSE (Java Secure Socket Extension) pour connecter les canaux de télémétrie à l'aide de SSL. Le démon pour dispositifs IBM WebSphere MQ Telemetry ne prend pas en charge SSL.

Configurez SSL pour authentifier le canal de télémétrie, le client MQTT et chiffrer le transfert des messages entre les clients et le canal de télémétrie.

Comme alternative à l'utilisation de SSL, certains types de réseaux privés virtuels (VPN), tels que IPsec, authentifient les noeuds finaux d'une connexion TCP/IP. Le réseau privé virtuel VPN chiffre chaque paquet transitant sur le réseau. Une fois qu'une telle connexion VPN est établie, vous avez établi un réseau sécurisé. Vous pouvez connecter les clients MQTT aux canaux de télémétrie avec TCP/IP sur le réseau VPN.

Vous pouvez configurer la connexion entre un client Java MQTT et un canal de télémétrie pour utiliser le protocole SSL sur TCP/IP. Les éléments sécurisé dépendent de la manière dont vous avez configuré SSL pour utiliser JSSE. En partant de la configuration la plus sécurisée, vous pouvez configurer trois niveaux différents de sécurité :

1. Autorisez uniquement les clients MQTT dignes de confiance à se connecter. Connectez un client MQTT uniquement à un canal de télémétrie digne de confiance. Chiffrez les messages entre le client et le gestionnaire de files d'attente ; voir «Authentification du client MQTT à l'aide de SSL», à la page 133
2. Connectez un client MQTT uniquement à un canal de télémétrie digne de confiance. Chiffrez les messages entre le client et le gestionnaire de files d'attente ; voir «Authentification du canal de télémétrie à l'aide de SSL», à la page 136.
3. Chiffrez les messages entre le client et le gestionnaire de files d'attente ; voir «Confidentialité de la publication sur les canaux de télémétrie», à la page 138.

Paramètres de configuration JSSE

Modifiez les paramètres JSSE afin de changer la façon dont une connexion SSL est configurée. Les paramètres de configuration JSSE sont composés de trois ensembles :

1. IBM WebSphere MQ Canal de télémétrie
2. client Java MQTT
3. JRE

Configurez les paramètres du canal de télémétrie à l'aide d'IBM WebSphere MQ Explorer. Définissez les paramètres du client Java MQTT dans l'attribut `MqttConnectionOptions.SSLProperties` . Modifiez les paramètres de sécurité en éditant les fichiers dans le répertoire de sécurité de JRE à la fois sur le client et sur le serveur.

IBM WebSphere MQ Canal de télémétrie

Définissez tous les paramètres SSL de canal de télémétrie à l'aide de WebSphere MQ Explorer.

`ChannelName`

`ChannelName` est un paramètre obligatoire sur tous les canaux.

Le nom de canal identifie le canal associé à un numéro de port particulier. Nommez les canaux de manière à vous aider à gérer des ensembles de clients MQTT.

PortNumber

PortNumber est un paramètre facultatif sur tous les canaux. La valeur par défaut est 1883 pour les canaux TCP et 8883 pour les canaux SSL.

Numéro de port TCP/IP associé à ce canal. Les clients MQTT sont connectés à un canal en indiquant le port défini pour le canal. Si le canal dispose des propriétés SSL, le client doit se connecter à l'aide du protocole SSL ; par exemple :

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

KeyFileName est un paramètre obligatoire pour les canaux SSL. Il doit être omis pour les canaux TCP.

KeyFileName est le chemin d'accès au magasin de clés Java contenant les certificats numériques que vous fournissez. Sur le serveur, utilisez les types de magasin de clés JKS, JCEKS ou PKCS12.

Identifiez le type de magasin de clés à l'aide de l'une des extensions de fichier suivantes :

```
.jks
.jceks
.p12
.pkcs12
```

Un magasin de clés associé à une autre extension est censé être un magasin JKS.

Vous pouvez combiner un magasin de clés d'un type donné se trouvant sur le serveur avec d'autres types de magasins de clés se trouvant sur le client.

Placez le certificat privé du serveur dans le magasin de clés. Le certificat est connu sous le nom de certificat serveur. Ce certificat peut être autosigné ou faire partie d'une chaîne de certificats signée par une autorité de signature.

Si vous utilisez une chaîne de certificats, placez les certificats associés dans le magasin de clés du serveur.

Le certificat du serveur et tous les certificats de la chaîne de certificats sont envoyés aux clients afin d'authentifier l'identité du serveur.

Si vous avez attribué à ClientAuth la valeur Required, le magasin de clés doit contenir tous les certificats nécessaires à l'authentification du client. Le client envoie un certificat autosigné, ou une chaîne de certificats, et est authentifié par la première vérification de cet élément par rapport à un certificat se trouvant dans le magasin de clés. Lorsque vous utilisez une chaîne de certificats, un certificat peut vérifier plusieurs clients, même s'ils sont émis avec des certificats client différents.

PassPhrase

PassPhrase est un paramètre obligatoire pour les canaux SSL. Il doit être omis pour les canaux TCP.

La phrase passe est utilisée pour protéger le magasin de clés.

ClientAuth

ClientAuth est un paramètre SSL facultatif. Par défaut, aucune authentification de client n'est effectuée. Il doit être omis pour les canaux TCP.

Définissez ClientAuth si vous voulez que le service de télémétrie (MQXR) authentifie le client avant de lui permettre de se connecter au canal de télémétrie.

Si vous définissez ClientAuth, le client doit se connecter au serveur à l'aide de SSL et authentifier le serveur. En réponse à la définition de ClientAuth, le client envoie son certificat numérique au serveur et tout autre certificat dans son magasin de clés. Son certificat numérique

est connu sous le nom de certificat client. Ces certificats sont authentifiés par rapport aux certificats se trouvant dans le magasin de clés du canal et dans le magasin `cacerts`.

CipherSuite

`CipherSuite` est un paramètre SSL facultatif. Par défaut, il essaie toutes les spécifications CipherSpecs activées. Il doit être omis pour les canaux TCP.

Si vous voulez utiliser une spécification CipherSpec particulière, attribuez à `CipherSuite` le nom de la spécification CipherSpec qui doit être utilisée pour établir la connexion SSL.

Le service de télémétrie et client MQTT négocient une spécification CipherSpec commune à partir de tous les CipherSpecs activés à chaque extrémité. Si un CipherSpec spécifique est indiqué à l'une ou aux deux extrémités, il doit correspondre au CipherSpec de l'autre extrémité.

Installez les chiffrements supplémentaires en ajoutant des fournisseurs supplémentaires à JSSE.

FIPS (Federal Information Processing Standards)

FIPS est un paramètre facultatif. Par défaut, il n'est pas défini.

Utilisez le panneau de propriété du gestionnaire de files d'attente ou **runmqsc**, pour définir SSLFIPS. SSLFIPS Indique si seuls les algorithmes certifiés pour FIPS doivent être utilisés.

Revocation namelist

Revocation namelist est un paramètre facultatif. Par défaut, il n'est pas défini.

Utilisez le panneau de propriété du gestionnaire de files d'attente ou **runmqsc**, pour définir SSLCRLNL. SSLCRLNL indique une liste de noms d'objet d'informations d'authentification utilisés pour fournir des emplacements de révocation de certificat.

Aucun autre gestionnaire de files d'attente qui a défini les propriétés SSL n'est utilisé.

Client Java MQTT

Définissez les propriétés SSL pour le client Java dans `MqttConnectionOptions.SSLProperties`; par exemple:

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

Les noms et les valeurs de propriétés spécifiques sont décrits dans la classe `MqttConnectOptions` . Pour des liens vers la documentation de l'API client pour les bibliothèques client MQTT, voir MQTT client programming reference.

Protocole

`Protocol` est facultatif.

Le protocole est sélectionné en négociation avec le serveur de télémétrie. Si vous avez besoin d'un protocole spécifique, vous pouvez en sélectionner un. Si le serveur de télémétrie ne prend pas en charge le protocole, la connexion échoue.

ContextProvider

`ContextProvider` est facultatif.

KeyStore

`KeyStore` est facultatif. Configurez-le si `ClientAuth` est défini au niveau du serveur pour forcer l'authentification du client.

Placez le certificat numérique du client, signé à l'aide de la clé privée, dans le magasin de clés. Spécifiez le chemin d'accès et le mot de passe du magasin de clés. Le type et le fournisseur sont facultatifs. JKS est le type par défaut et IBMJCE est le fournisseur par défaut.

Indiquez un fournisseur de magasin de clés pour référencer une classe qui ajoute un nouveau fournisseur de magasin de clés. Transmettez le nom de l'algorithme utilisé par le fournisseur de magasin de clés pour instancier `KeyManagerFactory` en définissant le nom du gestionnaire de clés.

`TrustStore`

`TrustStore` est facultatif. Vous pouvez placer tous les certificats que vous accréditez dans le magasin `cacerts` de JRE.

Configurez le magasin de clés de confiance si vous souhaitez disposer d'un fichier de ce type distinct pour le client. Vous pouvez ne pas configurer le magasin de clés de confiance si le serveur utilise un certificat émis par une autorité de certification connue dont le certificat racine est déjà stocké dans `cacerts`.

Ajoutez le certificat signé publiquement du serveur ou du certificat racine au magasin de clés de confiance, et indiquez le chemin d'accès et le mot de passe du magasin de clés de confiance. JKS est le type par défaut et IBMJCE est le fournisseur par défaut.

Indiquez un fournisseur de magasin de clés de confiance pour référencer une classe qui ajoute un nouveau fournisseur de magasin de clés de confiance. Transmettez le nom de l'algorithme utilisé par le fournisseur de magasin de clés de confiance pour instancier `TrustManagerFactory` en définissant le nom du gestionnaire d'accréditation.

JRE

D'autres aspects de la sécurité Java qui affectent le comportement de SSL côté client et côté serveur sont configurés dans le JRE. Les fichiers de configuration sur Windows se trouvent dans *Java Installation Directory*`\jre\lib\security`. Si vous utilisez le JRE livré avec IBM WebSphere MQ, le chemin est celui qui est indiqué dans le tableau suivant :

Tableau 8. Chemins de fichiers par plateforme pour les fichiers de configuration SSL JRE

Plateforme	**Chemin d'accès au fichier**
Windows	*WMQ Installation Directory*`\java\jre\lib\security`
Autres plateformes UNIX and Linux	*WMQ Installation Directory*`/java/jre64/jre/lib/security`

Autorités de certification reconnues

Le fichier `cacerts` contient les certificats racine des autorités de certification reconnues. Le fichier `cacerts` est utilisé par défaut, sauf si vous définissez un autre magasin de clés de confiance. Si vous utilisez le fichier `cacerts` ou que vous n'indiquez aucun magasin de clés de confiance, vous devez vérifier et éditer la liste des signataires du fichier `cacerts` pour remplir les conditions requises en matière de sécurité.

Vous pouvez ouvrir le fichier `cacerts` à l'aide de la commande WebSphere MQ `strmqikm` qui exécutez l'utilitaire IBM Key Management. Ouvrez le fichier `cacerts` en tant que fichier JKS avec le mot de passe `changeit`. Modifiez le mot de passe afin de sécuriser le fichier.

Configuration des classes de sécurité

Utilisez le fichier `java.security` pour enregistrer les fournisseurs de sécurité supplémentaires et d'autres propriétés de sécurité par défaut.

Droits d'accès

Utilisez le fichier `java.policy` pour modifier les droits d'accès accordés aux ressources. `javaws.policy` accorde des droits à `javaws.jar`

Puissance de chiffrement

Certains JRE sont livrés avec une puissance de chiffrement réduite. Si vous ne pouvez pas importer les clés dans les magasins de clés, la réduction de la puissance de chiffrement peut en être la cause. Démarrez **`ikeyman`** à l'aide de la commande **`strmqikm`**, ou téléchargez des

fichiers de juridiction de grande taille mais limités à partir du site IBM developer kits, Security information.

Important : Votre pays d'origine est peut-être assujetti à des restrictions relatives à l'importation, la possession, l'utilisation ou la réexportation de logiciel de chiffrement vers un autre pas. Avant de télécharger ou d'utiliser des fichiers de règles sans restriction, vous devez vérifier les lois applicables dans votre pays. Vérifiez ses règlements et sa politique en matière d'importation, de possession, d'utilisation et de réexportation de logiciel de chiffrement afin de déterminer si cette action est autorisée.

Modifiez le fournisseur d'accréditation pour permettre au client de se connecter à n'importe quel serveur.

L'exemple illustre comment ajouter un fournisseur d'accréditation et le référencer à partir du code du client MQTT. Dans l'exemple aucune authentification du client ou du serveur n'est effectuée. La connexion SSL résultante est chiffrée sans être authentifiée.

Le fragment de code dans la Figure 25, à la page 143 définit le fournisseur d'accréditation `AcceptAllProviders` et le gestionnaire d'accréditation pour le client MQTT.

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

Figure 25. Fragment de code pour le client MQTT

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
    private static final long serialVersionUID = 1L;
    public AcceptAllProvider() {
        super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
        put("TrustManagerFactory.TrustAllCertificates",
                AcceptAllTrustManagerFactory.class.getName());
    }
```

Figure 26. AcceptAllProvider.java

```
    protected static class AcceptAllTrustManagerFactory extends
            javax.net.ssl.TrustManagerFactorySpi {
        public AcceptAllTrustManagerFactory() {}
        protected void engineInit(java.security.KeyStore keystore) {}
        protected void engineInit(
                javax.net.ssl.ManagerFactoryParameters parameters) {}
        protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
            return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
        }
```

Figure 27. AcceptAllTrustManagerFactory.java

```
protected static class AcceptAllX509TrustManager implements
        javax.net.ssl.X509TrustManager {
    public void checkClientTrusted(
            java.security.cert.X509Certificate[] certificateChain,
            String authType) throws java.security.cert.CertificateException {
        report("Client authtype=" + authType);
        for (java.security.cert.X509Certificate certificate : certificateChain) {
            report("Accepting:" + certificate);
        }
    }
    public void checkServerTrusted(
            java.security.cert.X509Certificate[] certificateChain,
            String authType) throws java.security.cert.CertificateException {
        report("Server authtype=" + authType);
        for (java.security.cert.X509Certificate certificate : certificateChain) {
            report("Accepting:" + certificate);
        }
    }
    public java.security.cert.X509Certificate[] getAcceptedIssuers() {
        return new java.security.cert.X509Certificate[0];
    }
private static void report(String string) {
       System.out.println(string);
  }
}
```

Figure 28. AcceptAllX509TrustManager.java

Configuration JAAS du canal de télémétrie

Configurez JAAS pour authentifier le Nom d'`utilisateur` envoyé par le client.

l'administrateur WebSphere MQ configure les canaux MQTT qui nécessitent l'authentification du client à l'aide de JAAS. Indiquez le nom d'une configuration JAAS pour chaque canal qui doit effectuer une authentification JAAS. Les canaux peuvent tous utiliser la même configuration JAAS configuration, ou utiliser des configurations JAAS différentes. Les configurations sont définies dans *WMQData directory*\qmgrs*qMgrName*\mqxr\jaas.config.

Le fichier `jaas.config` est organisé par nom de configuration JAAS. Sous chaque nom de configuration figure une liste de configurations de connexion ; voir Figure 29, à la page 145.

JAAS fournit quatre normes de modules de connexion. Les modules de connexion standard NT et UNIX ont une valeur limitée.

JndiLoginModule
Effectue l'authentification sur un service de répertoire configuré sous JNDI (Java Naming and Directory Interface).

Krb5LoginModule
Effectue l'authentification à l'aide des protocoles Kerberos.

NTLoginModule
Effectue l'authentification à l'aide des informations de sécurité NT pour l'utilisateur en cours.

UnixLoginModule
Effectue l'authentification à l'aide des informations de sécurité UNIX pour l'utilisateur en cours.

Lorsque vous utilisez `NTLoginModule` ou `UnixLoginModule` le service de télémétrie (MQXR) s'exécute avec l'identité `mqm`, et non l'identité du canal MQTT. `mqm` est l'identité transmise à `NTLoginModule` ou `UnixLoginModule` pour l'authentification et non pas l'identité du client.

Pour résoudre ce problème, écrivez votre propre module de connexion ou utilisez d'autres modules de connexion standard. Un `JAASLoginModule.java` exemple est fourni avec WebSphere MQ Telemetry. Il s'agit d'une implémentation de l'interface `javax.security.auth.spi.LoginModule`. Utilisez-le pour développer votre propre méthode d'authentification.

Toute nouvelle classe LoginModule que vous fournissez doit figurer dans le chemin d'accès aux classes du service de télémétrie (MQXR). Ne placez pas vos classes dans les répertoires WebSphere MQ qui se trouvent dans le chemin d'accès aux classes. Créez vos propres répertoires et définissez le chemin d'accès complet aux classes pour le service de télémétrie (MQXR).

Vous pouvez accroître le chemin d'accès aux classes utilisé par le service de télémétrie (MQXR) en définissant le chemin d'accès aux classes dans le fichier `service.env`. CLASSPATH doit être indiqué en majuscules et peut uniquement contenir des littéraux. Vous ne pouvez pas utiliser de variables dans CLASSPATH ; par exemple, l'instruction CLASSPATH=%CLASSPATH% est incorrecte. Le service de télémétrie (MQXR) définit son propre chemin d'accès aux classes? La variable CLASSPATH définie dans le fichier `service.env` est ajoutée à ce dernier.

Le service de télémétrie (MQXR) fournit deux rappels qui renvoient le `nom d'utilisateur` et le `Mot de passe` pour le client connecté au canal MQTT. Le `nom d'utilisateur` et le `mot de passe` sont définis dans l'objet `MqttConnectOptions` . Voir la Figure 30, à la page 145 pour un exemple d'accès au `Nom d'utilisateur` et au `Mot de passe`.

Exemples

Modèle de fichier de configuration JAAS avec une configuration nommée, `MQXRConfig`.

```
MQXRConfig {
  samples.JAASLoginModule required debug=true;
  //com.ibm.security.auth.module.NTLoginModule required;
  //com.ibm.security.auth.module.Krb5LoginModule required
  //                 principal=principal@your_realm
  //                 useDefaultCcache=TRUE
  //                 renewTGT=true;
  //com.sun.security.auth.module.NTLoginModule required;
  //com.sun.security.auth.module.UnixLoginModule required;
  //com.sun.security.auth.module.Krb5LoginModule required
  //                 useTicketCache="true"
  //                 ticketCache="${user.home}${/}tickets";
};
```

Figure 29. Modèle de fichier `jaas.config`

Modèle de module de connexion JAAS codé pour recevoir le `Nom d'utilisateur` et le `Mot de passe` fournis par un client MQTT.

```
  public boolean login()
      throws javax.security.auth.login.LoginException {
    javax.security.auth.callback.Callback[] callbacks =
      new javax.security.auth.callback.Callback[2];
    callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
    callbacks[1] = new javax.security.auth.callback.PasswordCallback(
        "PasswordCallback", false);
    try {
      callbackHandler.handle(callbacks);
      String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
          .getName();
      char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
          .getPassword();
      // Accept everything.
      if (true) {
        loggedIn = true;
      } else
        throw new javax.security.auth.login.FailedLoginException("Login failed");

      principal= new JAASPrincipal(username);

    } catch (java.io.IOException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    }

    return loggedIn;
  }
```

Figure 30. Modèle de méthode `JAASLoginModule.Login()`

Concepts du démon pour périphériques IBM WebSphere MQ Telemetry

Le démon pour dispositifs IBM WebSphere MQ Telemetry est une application client MQTT V3 avancée. Utilisez-le pour stocker et transférer des messages à d'autres clients MQTT. Il se connecte à IBM WebSphere MQ comme un client MQTT, mais vous pouvez également y connecter d'autres clients MQTT.

Le démon est un courtier de publication/abonnement. Les clients MQTT V3 se connectent à lui pour publier sur des sujets ou s'y abonner, en utilisant des chaînes de sujet pour la publication, et des filtres de sujet pour les abonnements. La chaîne de sujet est hiérarchique, et les niveaux sont séparés par /. Les filtres de sujet sont des chaînes qui peuvent contenir des caractères génériques correspondant à un niveau +, et des caractères génériques correspondant à plusieurs niveaux # dans leur dernière portion.

Remarque : Les caractères génériques dans le démon suivent les règles plus restrictives de WebSphere Message Broker v6. IBM WebSphere MQ est différent. Il accepte plusieurs caractères génériques multiniveaux. Les caractères génériques peuvent représenter un nombre indéfini de niveaux à tout endroit de la chaîne de sujet.

Plusieurs clients MQTT v3 peuvent se connecter au démon à l'aide d'un port d'écoute. Le port d'écoute par défaut est le modifiable. Vous pouvez définir plusieurs ports d'écoute et leur affecter des espaces de noms distincts. Voir «Ports d'écoute du démon pour dispositifs WebSphere MQ Telemetry», à la page 154. Le démon est lui-même un client MQTT v3. Configurez une connexion de pont pour connecter le démon au port d'écoute d'un autre démon, ou à un service WebSphere MQ Telemetry (MQXR).

Vous pouvez configurer plusieurs ponts pour le démon pour dispositifs WebSphere MQ Telemetry. Utilisez les ponts pour connecter un réseau de démons échangeant des publications.

Chaque pont peut diffuser des publications et s'abonner aux sujets sur son démon local. Il peut aussi le faire sur un autre démon, un courtier de publication/abonnement WebSphere MQ, ou un autre courtier MQTT v3 auquel il est connecté. A l'aide d'un filtre de sujet, vous pouvez sélectionner les publications à propager d'un courtier à un autre. Vous pouvez propager les publications dans les deux sens. Vous pouvez propager les publications du démon local vers chacun de ses courtiers distants connectés, ou des courtiers connectés vers le démon local. Voir «Ponts du démon pour périphériques IBM WebSphere MQ Telemetry», à la page 146.

Ponts du démon pour périphériques IBM WebSphere MQ Telemetry

Un pont du démon pour dispositifs IBM WebSphere MQ Telemetry connecte deux courtiers de publication/abonnement à l'aide du protocole MQTT v3. Le pont propage les publications d'un courtier à un autre, dans n'importe quel sens. A l'une des extrémités se trouve la connexion de type pont du démon pour dispositifs WebSphere MQ Telemetry, et à l'autre extrémité peut se trouver un gestionnaire de files d'attente ou un autre démon. Un gestionnaire de files d'attente est connecté à la connexion de type pont par un canal de télémétrie. Un démon est connecté à la connexion de type pont à l'aide d'un programme d'écoute.

Le démon pour dispositifs IBM WebSphere MQ Telemetry prend en charge une ou plusieurs connexions simultanées à d'autres courtiers. Les connexions du démon s'appellent des ponts et sont définies par des entrées dans son fichier de configuration. Les connexions à IBM WebSphere MQ se font à l'aide de canaux de télémétrie IBM WebSphere MQ, comme le montre la figure qui suit :

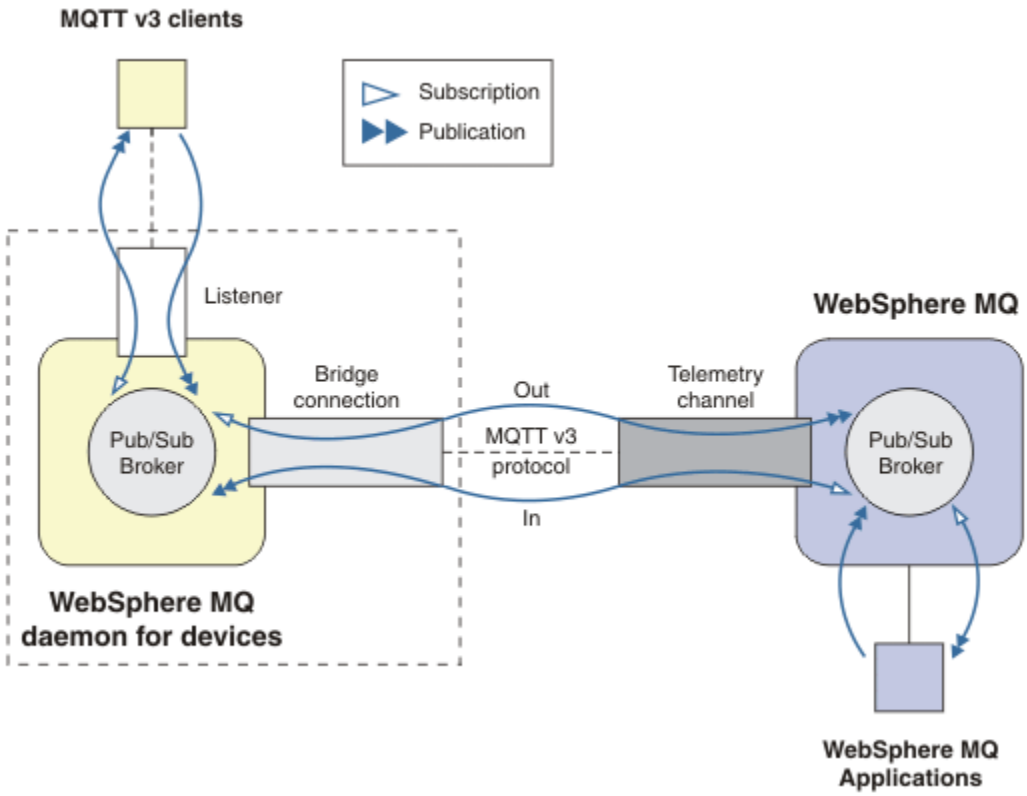


Figure 31. Connexion de IBM WebSphere MQ Telemetry daemon for devices à IBM WebSphere MQ

Un pont connecte le démon à un autre courtier en tant que client MQTT v3. Les paramètres du pont reproduisent les attributs d'un client MQTT v3.

Un pont est plus qu'une connexion. Il agit comme un agent de publication/abonnement situé entre deux courtiers de publication/abonnement. Le courtier local est le démon pour dispositifs IBM WebSphere MQ Telemetry et le courtier distant peut être n'importe quel courtier de publication/abonnement prenant en charge le protocole MQTT v3. Généralement, le courtier distant est un autre démon ou IBM WebSphere MQ.

Le rôle du pont est de propager les publications entre les deux courtiers. Le pont est bidirectionnel. Il propage les publications dans les deux directions. La figure Figure 31, à la page 147 illustre la manière dont le pont connecte le démon pour dispositifs IBM WebSphere MQ Telemetry à IBM WebSphere MQ. La rubrique «Exemples de paramétrage des sujets dans le pont», à la page 148 utilise des exemples pour illustrer l'utilisation du paramètre topic pour configurer le pont.

Les flèches In et Out de la Figure 31, à la page 147 indiquent la bidirectionnalité du pont. A une extrémité de la flèche, un abonnement est créé. Les publications qui lui correspondent sont publiées sur le courtier situé à l'extrémité opposée de la flèche. Le libellé de la flèche dépend du flux des publications. Les publications entrent dans le démon et en sortent. Les libellés sont importants car ils sont utilisés dans la syntaxe de commande. Gardez en mémoire qu'Entrée (In) et Sortie (Out) font référence au flux des publications, et non à la destination de l'abonnement.

D'autres clients, applications ou courtiers peuvent être connectés à IBM WebSphere MQ ou au démon pour dispositifs WebSphere MQ Telemetry. Ils diffusent des publications et s'abonnent aux sujets dans le courtier auquel ils sont connectés. Si le courtier est IBM WebSphere MQ, les sujets peuvent être groupés ou répartis, et non définis de façon explicite au niveau du gestionnaire de files d'attente local.

Utilisation des ponts

Connectez les démons entre eux à l'aide de connexions de type pont et de programmes d'écoute. Connectez les démons et les gestionnaires de files d'attente à l'aide de connexions de type pont et de canaux de télémétrie. Lorsque vous connectez plusieurs courtiers les uns aux autres, il est possible de créer des boucles. Prenez garde : les publications peuvent circuler sans fin sans être détectées le long d'une boucle de courtiers.

La connexion de démons à IBM WebSphere MQ via un pont peut être utilisée pour les raisons suivantes :

Réduire le nombre de connexions de clients MQTT à WebSphere MQ
En utilisant une hiérarchie de démons vous pouvez connecter de nombreux clients à WebSphere MQ. Plus que vous n'en pouvez connecter simultanément avec un unique gestionnaire de files d'attente.

Stocker et retransmettre les messages entre les clients MQTT et WebSphere MQ
Vous pouvez utiliser la fonction de stockage et de retransmission pour éviter de devoir conserver des connexions continues entre les clients et IBM WebSphere MQ, si les clients ne disposent pas de leur propre stockage. Vous pouvez utiliser plusieurs types de connexion entre le client MQTT et WebSphere MQ. Voir Concepts et scénarios de télémétrie pour la surveillance et le contrôle.

Filtrer les publications échangées entre les clients MQTT et WebSphere MQ
Généralement, les publications sont constituée en partie de messages traités en local, tandis que les autres messages impliquent d'autres applications. Les publications locales peuvent inclure des flux de contrôle entre les détecteurs et les actionneurs, et les publications distantes comprennent les demandes de lecture, les statuts et les commandes de configuration.

Modifier les espaces de sujet des publications
Il s'agit d'éviter les collisions entre les chaînes de sujet des clients connectés à différents ports d'écoute. Dans l'exemple, le démon affecte un libellé aux relevés de compteurs provenant des différents bâtiments. Voir Séparation des espaces de sujet des clients connectés à des démons différents.

Exemples de paramétrage des sujets dans le pont

Diffusion de toutes les publications au courtier distant - valeurs par défaut

Le sens par défaut est out (sortie), et le pont publie les sujets sur le courtier distant. Le paramètre topic contrôle les sujets qui sont propagés à l'aide de filtres.

Le pont utilise le paramètre topic dans Figure 32, à la page 148 pour s'abonner à tout ce qui est publié sur le démon local par les clients MQTT ou par d'autres courtiers. Il publie les sujets sur le courtier distant connecté par le pont.

```
connection Daemon1
topic #
```

Figure 32. Diffusion de toutes les publications au courtier distant

Diffusion de toutes les publications au courtier distant - valeurs explicites

Le paramètre topic dans le fragment de code suivant donne le même résultat que les valeurs par défaut. Le seule différence est que le paramètre **direction** est explicite. Utilisez le sens `out` (sortie) pour souscrire un abonnement au courtier local (le démon), et diffuser des publications sur le courtier distant. Les publications créées sur le démon local auquel le pont est abonné sont publiées sur le courtier distant.

```
connection Daemon1
topic # out
```

Figure 33. Diffusion de toutes les publications au courtier distant - valeurs explicites

Diffusion de toutes les publications au courtier local

Au lieu d'utiliser le sens `out` (sortie), vous pouvez définir le sens opposé, `in` (entrée). Le fragment de code suivant configure l'abonnement du pont à tout ce qui est publié sur le courtier distant connecté par le pont. Le pont publie les sujets sur le courtier local (le démon).

```
connection Daemon1
topic # in
```

Figure 34. Diffusion de toutes les publications au courtier local

Diffusion de toutes les publications du sujet export du courtier local dans le sujet import du courtier distant

Deux paramètres supplémentaires, **`local_prefix`** et **`remote_prefix`**, permettent de modifier le filtre de sujets, `#` dans les exemples suivants. L'un des paramètres sert à modifier le filtre de sujets utilisé dans l'abonnement, l'autre sert à modifier le sujet dans lequel la publication est diffusée. Le résultat recherché est de remplacer le début de la chaîne de sujet utilisée dans un courtier par une chaîne de sujet différente dans un autre courtier.

Selon le sens de la commande topic, la signification de **`local_prefix`** et de **`remote_prefix`** s'inverse. Si le sens est `out`, le paramètre par défaut **`local_prefix`** est utilisé pour l'abonnement au sujet, tandis que **`remote_prefix`** remplace le paramètre **`local_prefix`** de la chaîne de sujet dans la publication distante. Si le sens est `in`, **`remote_prefix`** est inséré dans l'abonnement distant, et **`local_prefix`** remplace le paramètre **`remote_prefix`** dans la chaîne de sujet.

La portion initiale d'une chaîne de sujet est souvent considérée comme l'élément de définition d'un espace de sujet. Les paramètres supplémentaires permettent de modifier l'espace dans lequel le sujet est publié. Vous pouvez ainsi empêcher les collisions potentielles entre le sujet propagé et un autre sujet sur le courtier cible, ou pour retirer une chaîne de sujet d'un point de montage.

Par exemple, dans le fragment de code suivant, toutes les publications diffusées dans la chaîne de sujet `export/#` dans le démon, sont republiées dans `import/#` dans le courtier distant.

```
topic # out export/ import/
```

Figure 35. Diffusion de toutes les publications du sujet export du courtier local dans le sujet import du courtier distant

Diffusion de toutes les publications du sujet export du courtier distant dans le sujet import du courtier local

Le fragment de code suivant montre la configuration inverse. Le pont s'abonne à tout ce qui est publié sur le serveur distant avec la chaîne de sujet `export/#`, et le publie sur le courtier local dans le sujet `import/#`.

```
connection Daemon1
topic # in import/ export/
```

Figure 36. Diffusion de toutes les publications du sujet export du courtier distant dans le sujet import du courtier local

Diffusion dans le courtier distant de toutes les publications du point de montage 1884/ avec les chaînes de sujet d'origine

Dans le fragment de code qui suit, le pont s'abonne à tout ce qui est publié par les clients connectés au point de montage 1884/ dans le démon local. Il diffuse au courtier distant toutes les publications du point de montage. La chaîne de point de montage 1884/ est retirée des sujets publiés sur le

serveur distant. Le paramètre *local_prefix* est identique à la chaîne de point de montage 1884/, et *remote_prefix* est une chaîne vide.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Figure 37. Diffusion dans le courtier distant de toutes les publications du point de montage 1884/ avec les chaînes de sujet d'origine

Séparation des espaces de sujet des clients connectés à des démons différents

Une application est écrite pour les compteurs d'énergie électrique afin de publier des relevés de compteur pour un bâtiment. Les relevés sont diffusés à l'aide de clients MQTT sur un démon hébergé dans le même bâtiment. Le sujet sélectionné pour la publications est `power`. La même application est déployée dans plusieurs bâtiments d'un complexe. Pour la surveillance du site et le stockage des données, les relevés de tous les bâtiments sont rassemblés à l'aide de connexions de type pont. Les connexions relient les démons des bâtiments à WebSphere MQ dans un site central.

Les applications client de chaque génération sont identiques, mais les données doivent être différenciées par génération. Chaque relevé comporte une rubrique `power` et doit être précédé du numéro de bâtiment pour pouvoir le distinguer. Le pont du premier bâtiment du complexe utilise le préfixe `meters/building01/`, celui du deuxième utilise le préfixe `meters/building02/`. Les relevés des autres bâtiments suivent le même modèle. WebSphere MQ reçoit les relevés avec des rubriques telles que `meters/building01/power`.

L'exemple est contriqué ; en pratique, l'espace de sujet dans lequel l'application est publiée est susceptible d'être configurable.

Le fichier de configuration de chaque démon contient une instruction de rubrique qui suit le modèle du fragment de code suivant :

```
connection Daemon1
topic power out "" meters/building01/
```

Figure 38. Séparation des espaces de sujet des clients connectés à des démons différents

Spécifiez une chaîne vide comme marque de réservation pour le paramètre local_prefix inutilisé.

Séparation des espaces de sujet des clients connectés au même démon

Supposons qu'un seul démon soit utilisé pour connecter tous les compteurs. Si l'application peut être configurée pour se connecter à différents ports, vous pouvez identifier les bâtiments en connectant le compteur de chaque bâtiment à un port d'écoute distinct, comme dans le fragment de code suivant. L'exemple factice illustre la manière dont des points de montage peuvent être utilisés.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Figure 39. Séparation des espaces de sujet des clients connectés au même démon

Remappage des différents sujets pour les publications transitant dans les deux sens

Dans la configuration du fragment de code suivant, le pont s'abonne à la rubrique unique b sur le courtier distant et transmet les publications relatives à b au démon local, en changeant la rubrique en a. Le pont s'abonne également à la rubrique unique x au niveau du courtier local et transfère les publications relatives à x au courtier distant, en remplaçant la rubrique par y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Figure 40. Remappage des différents sujets pour les publications transitant dans les deux sens

Il est à noter dans cet exemple que sur les deux courtiers, plusieurs sujets font l'objet d'un abonnement et reçoivent des publications. Sur les deux courtiers, les espaces de sujet sont disjoints.

Remappage des mêmes sujets pour les publications transitant dans les deux sens (boucle)

Contrairement à l'exemple précédent, la configuration de la Figure 41, à la page 151 produit généralement une boucle. Dans l'instruction topic `topic "" in a b`, le pont s'abonne à distance à b, et publie en local sur a. Dans l'autre instruction topic, le pont s'abonne en local à a, et publie à distance sur b. La même configuration peut s'écrire comme à la Figure 42, à la page 151.

Le résultat général est que si un client publie dans b à distance, la publication est transférée au démon local en tant que publication sur la rubrique a. Toutefois, lors de sa publication par le pont sur le démon local sur la rubrique a, la publication correspond à l'abonnement effectué par le pont sur la rubrique locale a. L'abonnement est `topic "" out a b`. Par conséquent, la publication est retransférée au courtier distant en tant que publication dans la rubrique b. Le pont est désormais abonné à la rubrique distante bet le cycle recommence.

Certains courtiers sont dotés d'un mécanisme de détection de boucle pour éviter ces situations. Mais ce mécanisme doit fonctionner lorsque différents types de courtiers sont interconnectés par des ponts. La détection de boucle ne fonctionne pas si WebSphere MQ est connecté par un pont au démon pour dispositifs WebSphere MQ Telemetry. En revanche, elle fonctionne si deux démons pour dispositifs IBM WebSphere MQ Telemetry sont connectés par un pont. La détection de boucle est activée par défaut. Voir la rubrique relative au paramètre try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

Figure 41. !Remappage des mêmes rubriques pour les publications circulant dans les deux sens

```
connection Daemon1
topic "" both a b
```

Figure 42. !Remappez les mêmes rubriques pour les publications circulant dans les deux sens, à l'aide de `both`.

La configuration de la Figure 40, à la page 151 est identique à celle de la Figure 41, à la page 151.

Disponibilité des connexions de type pont du IBM WebSphere MQ Telemetry daemon for devices

Configurez plusieurs adresses de connexion de pont pour le IBM WebSphere MQ Telemetry daemon for devices, pour lui permettre de se connecter au premier courtier distant disponible. Si le courtier est un

gestionnaire de files d'attente multi-instance, entrez ses deux adresses TCP/IP. Configurez une connexion principale pour le connecter ou le reconnecter au serveur principal, s'il est disponible.

La paramètre de la connexion de pont addresses est une liste d'adresses de connecteur TCP/IP. Le pont tente de se connecter successivement à chaque adresse, jusqu'à ce qu'il réussisse à établir une connexion. Les paramètres de connexion round_robin et start_type contrôlent la manière dont les adresses sont utilisées une fois que la connexion a été établie.

Si la valeur de start_type est auto, manual ou lazy, le pont tente de se reconnecter en cas de défaillance de la connexion. Il tente d'établir la connexion en suivant la séquence d'adresses, avec un délai de 20 secondes entre chaque tentative. Si la valeur de `start_type` est once, le pont ne tente pas de se reconnecter automatiquement si la connexion tombe.

Si la valeur de round_robin est true, le pont commence ses tentatives de connexion à la première adresse de la liste, puis les poursuit selon l'ordre de la liste. Lorsqu'il arrive à la fin de la liste, il recommence à la première adresse. Si la liste contient une seule adresse, il fait une tentative sur celle-ci toutes les 20 secondes.

Si la valeur de round_robin est false, la première adresse de la liste, qui est appelée le serveur principal, a la préférence. En cas d'échec de la première tentative de connexion au serveur principal, le pont continue d'essayer de s'y connecter en arrière-plan. En même temps, il tente de se connecter avec les autres adresses de la liste. Lorsque la tentative de connexion au serveur principal aboutit, le pont se déconnecte de la connexion active et passe sur la connexion au serveur principal.

Après une déconnexion volontaire, par exemple avec la commande **`connection_stop`**, le pont tente de réutiliser la même adresse au redémarrage de la connexion. Si la connexion est interrompue parce que le pont n'arrive pas à se connecter ou que le courtier distant perd la connexion, le pont attend 20 secondes. Il tente ensuite de se connecter à l'adresse qui suit dans la liste, ou à la même adresse si la liste n'en contient qu'une.

Connexion à un gestionnaire de files d'attente multi-instance

Dans une configuration de gestionnaire de files d'attente multi-instance, le gestionnaire de files d'attente s'exécute sur deux serveurs différents avec des adresses IP distinctes. Les canaux de télémétrie sont généralement configurés sans adresse IP spécifique. Ils ne sont configurés qu'avec un numéro de port. Au démarrage du canal de télémétrie, celui-ci sélectionne par défaut la première adresse réseau disponible sur le serveur local.

Configurez le paramètre addresses de la connexion de pont avec les deux adresses IP utilisées par le gestionnaire de files d'attente. Affectez à round_robin la valeur true.

En cas de défaillance de l'instance active du gestionnaire de files d'attente, le gestionnaire de files d'attente bascule sur l'instance de secours. Le démon détecte l'arrêt de la connexion à l'instance active et tente de se reconnecter à l'instance de secours. Il utilise la deuxième adresse Ip dans la liste des adresses configurées pour la connexion de pont.

Le pont est toujours connecté au même gestionnaire de files d'attente. Le gestionnaire de files d'attente récupère son état. Si cleansession a la valeur false, la session de la connexion de pont est restaurée à l'état qu'elle avait avant la défaillance. La connexion est rétablie au bout d'un certain temps. Des messages avec la qualité de service "au moins une fois" ou "au plus une fois" ne sont pas perdus et les abonnements continuent à fonctionner.

La durée de la reconnexion dépend du nombre de canaux et de clients qui redémarrent au démarrage de l'instance de secours, et au nombre de messages en cours. Le pont peut faire un certain nombre de tentatives sur les deux adresse IP avant de réussir à rétablir la connexion.

Ne configurez pas le canal de télémétrie d'un gestionnaire de files d'attente multi-instance avec une adresse IP spécifique. L'adresse IP n'est valide que sur un serveur.

La configuration d'un canal de télémétrie avec une adresse IP spécifique peut être souhaitable si vous utilisez une autre solution à haute disponibilité qui gère l'adresse IP.

cleansession

Une connexion de pont est une session du client MQTT v3. Vous pouvez choisir si la connexion démarre une nouvelle session ou si elle en restaure une existante. Si elle restaure une session existante, la connexion de pont préserve les abonnements et les publications conservées de la session précédente.

Ne définissez pas cleansession sur false si adresses répertorie plusieurs adresses IP et que les adresses IP se connectent à des canaux de télémétrie hébergés par différents gestionnaires de files d'attente ou à différents démons de télémétrie. L'état de la session n'est pas transféré entre les gestionnaires de files d'attente et les démons. Une tentative de redémarrage d'une session existante sur un autre gestionnaire de files d'attente ou un autre démon aboutit au lancement d'une nouvelle session. Les messages en attente de validation sont perdus, et les abonnements peuvent se comporter de façon inattendue.

Notifications

Les notifications permettent aux applications de savoir si la connexion de pont est active. Une notification est une publication dont la valeur est `1` (connecté), ou `0` (déconnecté). Elle est publiée dans la *chaîne de sujet* définie par le paramètre `notification_topic`. La valeur par défaut de *topicString* est $SYS/`broker/connection/` *clientIdentifier*`/state`. La valeur par défaut *topicString* contient le préfixe $SYS. Créez un abonnement aux sujets commençant par $SYS en définissant un filtre de sujet commençant par $SYS. Le sujet de filtre # crée sur le démon un abonnement à tout, sauf aux sujets commençant par $SYS. Pensez à $SYS comme à la définition d'un espace de sujet système spécial distinct de l'espace de sujet de l'application.

Les notifications permettent à IBM WebSphere MQ Telemetry daemon for devices de notifier les clients MQTT lorsqu'un pont est connecté ou déconnecté.

keepalive_interval

Le paramètre de connexion de pont keepalive_interval définit l'intervalle entre les commandes ping TCP/IP envoyées par le pont au serveur distant. L'intervalle par défaut est 60 secondes. La commande ping empêche le serveur distant ou un pare-feu détectant une période d'inactivité de fermer la session TCP/IP.

clientId

Une connexion de pont est une session client MQTT v3 dont le paramètre `clientIdentifier` est défini par le paramètre de connexion de pont clientid. Si vous souhaitez que les reconnexions reprennent une session précédente en affectant au paramètre cleansession la valeur false, l'attribut `clientIdentifier` utilisé dans chaque session doit être le même. La valeur par défaut de clientid est *hostname.connectionName* et reste la même.

Installation, vérification, configuration et contrôle du démon pour dispositifs WebSphere MQ Telemetry

L'installation, la vérification, la configuration et le contrôle du démon se font à partir de fichiers.

Installez le démon en copiant le kit de développement de logiciels sur le périphérique sur lequel vous allez exécuter le démon .

Par exemple, exécutez l'utilitaire client MQTT et connectez-vous au démon pour dispositifs WebSphere MQ Telemetry en tant que courtier de publication / abonnement ; voir Publier un message sur un client MQTT v3 spécifique .

Configurez le démon en créant un fichier de configuration. Voir Fichier de configuration du démon pour dispositifs WebSphere MQ.

Contrôlez un démon actif en créant des commandes dans le fichier `amqtdd.upd`. Toutes les 5 secondes, le démon lit le fichier, exécute les commandes et supprime le fichier. Voir Fichier de commandes du démon pour dispositifs WebSphere MQ Telemetry.

Ports d'écoute du démon pour dispositifs WebSphere MQ Telemetry

Connectez les clients MQTT V3 au démon pour dispositifs WebSphere MQ Telemetry à l'aide des ports d'écoute. Un point de montage et un nombre maximal de connexions peuvent être définis pour les ports d'écoute.

Un port d'écoute doit correspondre au numéro de port défini dans la méthode `connect(serverURI)` du client MQTT qui s'y connecte. Sur le client, comme sur le démon, sa valeur par défaut est 1883.

Vous pouvez modifier le port par défaut du démon en définissant le paramètre global `port` dans son fichier de configuration. Vous pouvez définir des ports spécifiques en ajoutant une définition `listener` au fichier de configuration du démon.

Pour chaque port d'écoute autre que le port par défaut, vous pouvez définir un point de montage pour isoler les clients. Les clients connectés à un port avec un point de montage sont isolés des autres clients. Voir «Points de montage du démon pour dispositifs WebSphere MQ Telemetry», à la page 154.

Vous pouvez limiter le nombre de clients qui peuvent se connecter à un port. Utilisez la définition globale `max_connections` pour limiter les connexions au port par défaut, ou qualifier caque port d'écoute à l'aide de `max_connections`.

Exemple

Exemple de fichier de configuration qui remplace le port par défaut 1883 par 1880 et limite à 10000 le nombre de connexions au port 1880. Le nombre de connexions au port 1884 est limité à 1000. Les clients connectés au port 1884 sont isolés de ceux qui sont connectés aux autres ports.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Points de montage du démon pour dispositifs WebSphere MQ Telemetry

Vous pouvez associer un point de montage à un port d'écoute utilisé par les clients MQTT pour les connecter à un démon pour dispositifs WebSphere MQ Telemetry. Un point de montage isole les publications et les abonnements échangés par les clients MQTT utilisant un port d'écoute des clients MQTT connectés à un autre port d'écoute.

Les clients connectés à un port d'écoute avec un point de montage ne peuvent jamais échanger directement des sujets avec les clients connectés à d'autres ports d'écoute. Les clients connectés à un port d'écoute sans point de montage peuvent diffuser des publications ou créer des abonnements sur les sujets de tous les clients. Les clients ne détectent pas qu'ils sont connectés par un point de montage. Cela n'a pas d'incidence sur les chaînes de sujet qu'ils créent.

Un point de montage est une chaîne de texte qui est ajoutée en préfixe aux chaînes de sujet des publications et des abonnements. Il est ajouté en préfixe à toutes les chaînes de sujet créées par les clients connectés au port d'écoute avec un point de montage. Cette chaîne est retirée de toutes les chaînes de sujet envoyées aux clients connectés au port d'écoute.

Si un port d'écoute n'a pas de point de montage, les chaînes de sujet des publications et des abonnements créés et reçus par les clients connectés au port ne sont pas modifiées.

Ajoutez / à la fin des chaînes de point de montage que vous créez. De cette manière, le point de montage est le sujet parent de l'arborescence de sujets pour le point de montage.

Exemple

Un fichier de configuration contient les ports d'écoute suivants :

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
```

```
mount_point 1884/
listener 1885
```

Un client, connecté au port 1883, crée un abonnement à `MyTopic`. Le démon enregistre l'abonnement comme `1883/MyTopic`. Un autre client connecté au port 1883 publie un message sur le sujet `MyTopic`. Le démon remplace la chaîne de sujet par `1883/MyTopic` et recherche des abonnements correspondants. L'abonné du port 1883 reçoit la publication avec sa chaîne de sujet d'origine `MyTopic`. Le démon a retiré le préfixe du point de montage de la chaîne de sujet.

Un autre client, connecté au port 1884, publie également sur le sujet `MyTopic`. Cette fois, le démon enregistre le sujet comme `1884/MyTopic`. L'abonné du port 1883 ne reçoit pas la publication, car différents points de montage aboutissent à un abonnement avec des chaînes de sujet différentes.

Un client, connecté au port 1885, publie sur le sujet `1883/MyTopic`. Le démon ne modifie pas la chaîne de sujet. L'abonné du port 1883 reçoit la publication sur le sujet `MyTopic`.

Qualité du service du démon pour dispositifs WebSphere MQ Telemetry, abonnements durables et publications conservées

Les paramètres de qualité du service s'appliquent à un seul démon en cours d'exécution. Si le démon s'arrête, soit de manière contrôlée, ou à cause d'une défaillance, l'état des messages en cours est perdu. La distribution d'un message au moins une fois, ou au plus une fois, ne peut pas être garantie en cas d'arrêt du démon. Le démon pour dispositifs WebSphere MQ Telemetry prend en charge une persistance limitée. Pour enregistrer les publications conservées et les abonnements en cas d'arrêt du démon, définissez le paramètre de configuration **`retained_persistence`**.

Contrairement à WebSphere MQ, le démon pour dispositifs WebSphere MQ Telemetry ne consigne pas les données persistantes. Ni l'état de la session et des messages, ni les publications conservées ne sont enregistrés dans les transactions. Par défaut, le démon supprime toutes les données lorsqu'il s'arrête. Une option permet de faire des points de contrôle réguliers sur les abonnements et les publications conservées. Le statut des messages est toujours perdu à l'arrêt du démon. Toutes les publications non conservées le sont aussi.

Définissez l'option de configuration du démon `Retained_persistence` avec la valeur `true` pour enregistrer périodiquement dans un fichier les publications conservées. Au redémarrage du démon, le dernier enregistrement automatique des publications conservées est rétabli. Par défaut, les messages conservés créés par les clients ne sont pas rétablis au redémarrage du démon.

Définissez l'option de configuration du démon `Retained_persistence` avec la valeur `true` pour enregistrer périodiquement dans un fichier les abonnements créés dans une session persistante. Si `Retained_persistence` a la valeur `true`, les abonnements créés par les clients dans une session dans laquelle `CleanSession` a la valeur `false`, donc une "session persistante", sont restaurés. Le démon restaure les abonnements lorsqu'il redémarre, ce qui déclenche la réception des publications. Le client reçoit les publications lorsqu'il redémarre avec `CleanSession` et la valeur `false`. Par défaut, l'état de la session client n'est pas enregistré à l'arrêt d'un démon. Les abonnements ne sont donc pas restaurés, même si le client définit `CleanSession` à la valeur `false`.

`Retained_persistence` est un mécanisme d'enregistrement automatique. Il n'enregistre pas forcément les dernières publications conservées ou les derniers abonnements. Vous pouvez modifier la fréquence d'enregistrement des publications conservées et des abonnements. Définissez l'intervalle ou le nombre de modifications entre les enregistrements, à l'aide des options de configuration `autosave_on_changes` et `autosave_interval`.

Exemple de configuration de la persistance

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
port 1882
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

Sécurité du démon pour dispositifs WebSphere MQ Telemetry

Le démon pour dispositifs WebSphere MQ Telemetry peut authentifier les clients qui se connectent à lui, utiliser des données d'identification pour se connecter à d'autres courtiers et contrôler l'accès aux sujets. La sécurité fournie par le démon est limitée par le fait qu'elle est construite à l'aide du client WebSphere MQ Telemetry pour C, qui ne prend pas en charge SSL. En conséquence, les connexions en entrée et en sortie du démon ne sont pas chiffrées, et ne peuvent pas être authentifiées à l'aide de certificats.

Par défaut, aucune sécurité n'est activée.

Authentification des clients

Les clients MQTT peuvent définir un nom d'utilisateur et un mot de passe à l'aide des méthodes `MqttConnectOptions.setUserName` et `MqttConnectOptions.setPassword`.

Authentifiez un client qui se connecte au démon en vérifiant le nom d'utilisateur et le mot de passe qu'il fournit par rapport aux entrées du fichier de mots de passe. Pour activer l'authentification, créez un fichier de mots de passe et définissez le paramètre `password_file` dans le fichier de configuration du démon. Voir password_file.

Définissez la paramètre `allow_anonymous` dans le fichier de configuration du démon pour permettre aux clients qui se connectent sans nom d'utilisateur et mot de passe de se connecter à un démon qui vérifie l'authentification. Voirallow_anonymous. Lorsqu'un client fournit un nom d'utilisateur et un mot de passe, celui-ci est toujours vérifié dans le fichier de mots de passe si le paramètre `password_file` est défini.

Définissez le paramètre `clientid_prefixes` dans le fichier de configuration du démon pour limiter la connexion à des clients spécifiques. Les clients doivent avoir un `clientIdentifiers` commençant par l'un des préfixes listés dans le paramètre `clientid_prefixes`. Voir clientid_prefixes.

Sécurité de la connexion de pont

Chaque connexion de type pont du démon pour dispositifs WebSphere MQ Telemetry est un client MQTT V3. Vous pouvez définir le nom d'utilisateur et le mot de passe de chaque connexion de pont en tant que paramètre dans le fichier de configuration du démon. Voir username etpassword. Le pont peut alors s'authentifier auprès du courtier.

Contrôle d'accès des sujets

Si les clients sont authentifiés, le démon peut également fournir le contrôle d'accès aux sujets pour chaque utilisateur. Il accorde l'accès sur la base de la correspondance entre le sujet sur lequel le client diffuse une publication ou crée un abonnement et une chaîne de sujet du fichier de contrôle d'accès. Voir acl_file.

La liste de contrôle d'accès contient deux parties. La première partie contrôle l'accès de tous les clients, y compris celui des clients anonymes. La seconde partie contient une section réservée aux utilisateurs du fichier de mots de passe. Elle liste les accès spécifiques des utilisateurs individuels.

Exemple

L'exemple qui suit montre les paramètres de sécurité.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Figure 43. Fichier de configuration du démon

```
Fred:Fredpassword
Barney:Barneypassword
```

Figure 44. Fichier de mot de passe passwords.txt

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

Figure 45. Fichier de contrôle d'accès acl.txt

Administration de la multidiffusion

Utilisez ces informations pour en savoir plus sur les tâches d'administration de multidiffusion WebSphere MQ , telles que la réduction de la taille des messages de multidiffusion et l'activation de la conversion de données.

Initiation à la multidiffusion

Utilisez ces informations pour vous initier aux rubriques de multidiffusion WebSphere MQ et aux objets d'informations de communication.

Pourquoi et quand exécuter cette tâche

WebSphere MQ La messagerie multidiffusion utilise le réseau pour distribuer des messages en mappant des rubriques à des adresses de groupe. Les tâches suivantes permettent de tester rapidement si l'adresse IP et le port requis sont correctement configurés pour la messagerie multidiffusion.

Création d'un objet COMMINFO pour la multidiffusion

L'objet informations de communication (COMMINFO) contient les attributs associés à la transmission multidiffusion. Pour plus d'informations sur les paramètres de l'objet COMMINFO, voir DEFINE COMMINFO.

Utilisez l'exemple de ligne de commande suivant pour définir un objet COMMINFO pour la multidiffusion:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

où *MC1* est le nom de votre objet COMMINFO, *adresse de groupe* est l'adresse IP de multidiffusion de votre groupe ou le nom DNS, et le *numéro de port* est le port sur lequel effectuer la transmission (la valeur par défaut est 1414).

Un nouvel objet COMMINFO appelé *MC1* est créé. Il s'agit du nom que vous devez spécifier lors de la définition d'un objet TOPIC dans l'exemple suivant.

Création d'un objet TOPIC pour la multidiffusion

Une rubrique est l'objet des informations publiées dans un message de publication/abonnement, et une rubrique se définit en créant un objet TOPIC. Les objets TOPIC ont deux paramètres qui définissent s'ils peuvent être utilisés avec la multidiffusion ou non. Ces paramètres sont: **COMMINFO** et **MCAST**.

- **COMMINFO** Ce paramètre indique le nom de l'objet d'information de communication multidiffusion. Pour plus d'informations sur les paramètres de l'objet COMMINFO, voir DEFINE COMMINFO.

- **MCAST** Ce paramètre indique si la multidiffusion est autorisée à ce niveau dans l'arborescence des rubriques.

Utilisez l'exemple de ligne de commande suivant pour définir un objet TOPIC pour la multidiffusion:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

Un nouvel objet TOPIC appelé *ALLSPORTS* est créé. Il comporte une chaîne de rubrique *Sports*, son objet d'informations de communication associé est appelé *MC1* (nom que vous avez spécifié lors de la définition d'un objet COMMINFO dans l'exemple précédent) et la multidiffusion est activée.

Test de la publication / abonnement multidiffusion

Une fois les objets TOPIC et COMMINFO créés, ils peuvent être testés à l'aide de l'exemple `amqspubc` et de l'exemple `amqssubc` . Pour plus d'informations sur ces exemples, voir Exemples de programmes de publication / abonnement.

1. Ouvrez deux fenêtres de ligne de commande. La première ligne de commande concerne l'exemple de publication `amqspubc` et la deuxième ligne de commande concerne l'exemple d'abonnement `amqssubc` .
2. Entrez la commande suivante sur la ligne de commande 1:

   ```
   amqspubc Sports QM1
   ```

   où *Sports* est la chaîne de rubrique de l'objet TOPIC défini dans un exemple précédent, et *QM1* est le nom du gestionnaire de files d'attente.
3. Entrez la commande suivante sur la ligne de commande 2:

   ```
   amqssubc Sports QM1
   ```

   où *Sports* et *QM1* sont identiques à ceux utilisés à l'étape «2», à la page 158.
4. Entrez `Hello world` sur la ligne de commande 1. Si le port et l'adresse IP spécifiés dans l'objet COMMINFO sont configurés correctement, l'exemple `amqssubc` , qui écoute sur le port les publications à partir de l'adresse spécifiée, génère `Hello world` sur la ligne de commande 2.

Topologie de rubrique de multidiffusion IBM WebSphere MQ

Utilisez cet exemple pour comprendre la topologie de rubrique IBM WebSphere MQ Multicast.

La prise en charge de la multidiffusion IBM WebSphere MQ requiert que chaque sous-arborescence ait son propre groupe de multidiffusion et son propre flux de données au sein de la hiérarchie totale.

Le schéma d'adressage IP *réseau avec classe* désigne un espace adresse pour l'adresse de multidiffusion. La plage de multidiffusion complète d'adresses IP est `224.0.0.0` à `239.255.255.255`, mais certaines de ces adresses sont réservées. Pour obtenir la liste des adresses réservées, contactez votre administrateur système ou consultez IPv4 Multicast Address Space Registry pour plus d'informations. Il est recommandé d'utiliser l'adresse de multidiffusion locale dans la plage de `239.0.0.0` à `239.255.255.255`.

Dans le diagramme suivant, il existe deux flux de données de multidiffusion possibles:

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

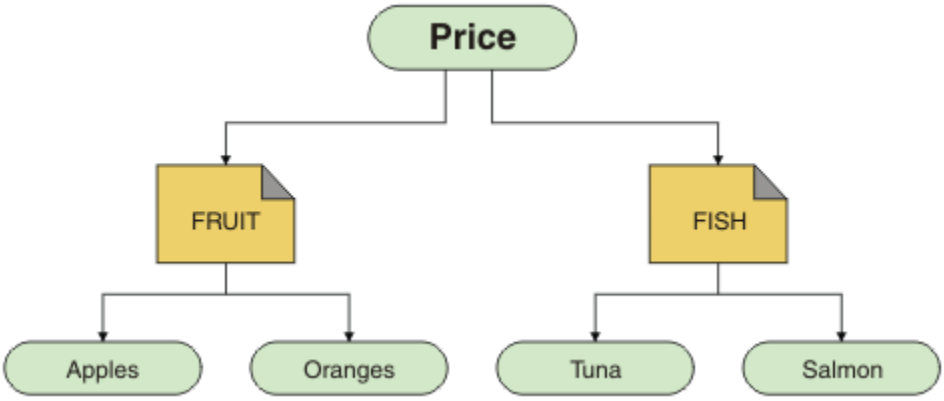
où *239.XXX.XXX.XXX* et *239.YYY.YYY.YYY* sont des adresses de multidiffusion valides.

Ces définitions de rubrique sont utilisées pour créer une arborescence de rubriques, comme illustré dans le diagramme suivant:

```

DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
```

```
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Chaque objet d'informations de communication multidiffusion (COMMINFO) représente un flux de données différent car leurs adresses de groupe sont différentes. Dans cet exemple, la rubrique FRUIT est définie pour utiliser l'objet COMMINFO MC1, la rubrique FISH est définie pour utiliser l'objet COMMINFO MC2et le noeud Price n'a pas de définitions de multidiffusion.

WebSphere MQ La multidiffusion est limitée à 255 caractères pour les chaînes de rubrique. Cette limitation signifie que les noms des noeuds et des noeuds feuille doivent être pris en compte dans l'arborescence. Si les noms des noeuds et des noeuds feuille sont trop longs, la chaîne de rubrique peut dépasser 255 caractères et renvoyer le code anomalie 2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR . Il est recommandé de rendre les chaînes de rubrique aussi courtes que possible car des chaînes de rubrique plus longues peuvent avoir un effet négatif sur les performances.

Contrôle de la taille des messages de multidiffusion

Utilisez ces informations pour en savoir plus sur le format de message WebSphere MQ et réduire la taille des messages WebSphere MQ .

Les messages WebSphere MQ sont associés à un certain nombre d'attributs contenus dans le descripteur de message. Pour les messages de petite taille, ces attributs peuvent représenter la majeure partie du trafic de données et peuvent avoir un effet négatif significatif sur le débit de transmission. WebSphere MQ La multidiffusion permet à l'utilisateur de configurer les attributs qui, le cas échéant, sont transmis avec le message.

La présence d'attributs de message, autres que la chaîne de rubrique, varie selon que l'objet COMMINFO indique qu'ils doivent être envoyés ou non. Si un attribut n'est pas transmis, l'application de réception applique une valeur par défaut. Les valeurs MQMD par défaut ne sont pas nécessairement identiques à la valeur MQMD_DEFAULT et sont décrites dans Tableau 9, à la page 160.

L'objet COMMINFO contient l'attribut MCPROP qui contrôle le nombre de zones MQMD et de propriétés utilisateur qui circulent avec le message. En définissant la valeur de cet attribut sur un niveau approprié, vous pouvez contrôler la taille des messages de multidiffusion WebSphere MQ :

MCPROP
Les propriétés de multidiffusion contrôlent la quantité de propriétés MQMD et propriétés utilisateur émises avec le message.

TOUT
Toutes les propriétés utilisateur et toutes les zones du MQMD sont transmises.

REPONSE
Seules les propriétés utilisateur et les zones MQMD liées à la réponse des messages sont transmises. Ces propriétés sont :

- MsgType
- MessageId

- CorrelId
- ReplyToQ
- ReplyToQmgr

UTILISATEUR
Seules les propriétés utilisateur sont transmises.

AUCUN
Aucune propriété utilisateur ni zone MQMD n'est transmise.

COMPAT
Cette valeur entraîne la transmission du message en mode compatible à RMM, ce qui permet une certaine interopérabilité avec les applications XMS en cours et les applications WebSphere Message Broker RMM .

Attributs de message de multidiffusion

Les attributs de message peuvent provenir de différents emplacements, tels que MQMD, les zones de MQRFH2et les propriétés de message.

Le tableau suivant montre ce qui se passe lorsque des messages sont envoyés en fonction de la valeur de MCPROP et de la valeur par défaut utilisée lorsqu'un attribut n'est pas envoyé.

Tableau 9. Attributs de messagerie et comment ils sont liés à la multidiffusion

Attribut	Action lors de l'utilisation de la multidiffusion	Valeur par défaut si non transmise
TopicString	Toujours inclus	Non applicable
MQMQ StrucId	Non transmis	Non applicable
Version MQMD	Non transmis	Non applicable
Rapport	Inclus si non par défaut	0
MsgType	Inclus si non par défaut	MQMT_DATAGRAM
Expiration	Inclus si non par défaut	0
Commentaires	Inclus si non par défaut	0
Codage	Inclus si non par défaut	MQENC_NORMAL (équiv)
CodedCharSetId	Inclus si non par défaut	1208
Format	Inclus si non par défaut	MQRFH2
Priorité	Inclus si non par défaut	4
Persistance	Inclus si non par défaut	MQPER_NON_PERSISTENT
MsgId	Inclus si non par défaut	Null
CorrelId	Inclus si non par défaut	Null
BackoutCount	Inclus si non par défaut	0
ReplyToQ	Inclus si non par défaut	Blanc
ReplyToQMgr	Inclus si non par défaut	Blanc
UserIdentifier	Inclus si non par défaut	Blanc
AccountingToken	Inclus si non par défaut	Null
PutAppIType	Inclus si non par défaut	MQAT_JAVA

Tableau 9. Attributs de messagerie et comment ils sont liés à la multidiffusion (suite)		
Attribut	**Action lors de l'utilisation de la multidiffusion**	**Valeur par défaut si non transmise**
PutAppIName	Inclus si non par défaut	Blanc
PutDate	Inclus si non par défaut	Blanc
PutTime	Inclus si non par défaut	Blanc
ApplOriginData	Inclus si non par défaut	Blanc
GroupID	Exclu	Non applicable
MsgSeqNumber	Exclu	Non applicable
Décalage	Exclu	Non applicable
MsgFlags	Exclu	Non applicable
OriginalLength	Exclu	Non applicable
UserProperties	Inclus	Non applicable

Référence associée
ALTER COMMINFO
DEFINI COMMINFO

Activation de la conversion de données pour la messagerie multidiffusion

Utilisez ces informations pour comprendre le fonctionnement de la conversion de données pour la messagerie multidiffusion WebSphere MQ .

WebSphere MQ La multidiffusion est un protocole partagé et sans connexion. Par conséquent, il n'est pas possible pour chaque client d'effectuer des demandes spécifiques de conversion de données. Chaque client abonné au même flux de multidiffusion reçoit les mêmes données binaires ; par conséquent, si une conversion de données WebSphere MQ est requise, la conversion est effectuée localement sur chaque client.

Dans une installation à plateforme mixte, il se peut que la plupart des clients requièrent les données dans un format qui n'est pas le format natif de l'application de transmission. Dans ce cas, les valeurs **CCSID** et **ENCODING** de l'objet COMMINFO de multidiffusion peuvent être utilisées pour définir le codage de la transmission de message à des fins d'efficacité.

WebSphere MQ La multidiffusion prend en charge la conversion de données de la charge de message pour les formats intégrés suivants:

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

En plus de ces formats, vous pouvez également définir vos propres formats et utiliser un exit de conversion de données MQDXP-Paramètre d'exit de conversion de données .

Pour plus d'informations sur la programmation des conversions de données, voir Conversion de données dans MQI pour la messagerie multidiffusion.

Pour plus d'informations sur la conversion de données, voir Conversion de données.

Pour plus d'informations sur les exits de conversion de données et `ClientExitPath`, voir la section ClientExitPath du fichier de configuration client.

Surveillance des applications de multidiffusion

Utilisez ces informations pour en savoir plus sur l'administration et la surveillance de WebSphere MQ Multicast.

L'état des diffuseurs actuels et des abonnés pour le traffic multidiffusion (par example, le nombre de messages envoyés et reçus, ou le nombre de messages perdus) est périodiquement transmis au serveur depuis le client. Lorsque le statut est reçu, l'attribut COMMEV de l'objet COMMINFO indique si le gestionnaire de files d'attente insère ou non un message d'événement dans SYSTEM.ADMIN.PUBSUB.EVENT. Le message d'événement contient les informations de statut reçues. Ces informations sont un outil de diagnostic précieux pour trouver la source d'un problème.

Utilisez la commande MQSC **DISPLAY CONN** pour afficher les informations de connexion relatives aux applications connectées au gestionnaire de files d'attente. Pour plus d'informations sur la commande **DISPLAY CONN** , voir DISPLAY CONN.

Utilisez la commande MQSC **DISPLAY TPSTATUS** pour afficher le statut de vos diffuseurs de publications et de vos abonnés. Pour plus d'informations sur la commande **DISPLAY TPSTATUS** , voir DISPLAY TPSTATUS.

COMMEV et l'indicateur de fiabilité des messages multidiffusion

L' *indicateur de fiabilité*, utilisé conjointement avec l'attribut **COMMEV** de l'objet COMMINFO, est un élément clé de la surveillance des diffuseurs de publications et des abonnés WebSphere MQ Multicast. L'indicateur de fiabilité (zone **MSGREL** renvoyée dans les commandes de statut de publication ou d'abonnement) est un indicateur WebSphere MQ qui illustre le pourcentage de transmissions qui ne comportent pas d'erreurs. Parfois, les messages doivent être retransmis en raison d'une erreur de transmission, ce qui se reflète dans la valeur de **MSGREL**. Les causes potentielles des erreurs de transmission incluent les abonnés lents, les réseaux occupés et les indisponibilités du réseau. **COMMEV** contrôle si les messages d'événement sont générés pour les descripteurs de multidiffusion créés à l'aide de l'objet COMMINFO et sont définis sur l'une des trois valeurs possibles:

DESACTIVE
Les messages d'événement ne sont pas écrits.

Activée
Les messages d'événement sont toujours écrits, avec une fréquence définie dans le paramètre COMMINFO **MONINT** .

EXCEPTION
Les messages d'événement sont enregistrés si la fiabilité des messages est inférieure au seuil de fiabilité. Un niveau de fiabilité de message inférieur ou égal à 90% indique qu'il peut y avoir un problème avec la configuration réseau ou qu'une ou plusieurs applications de publication / abonnement s'exécutent trop lentement:

- La valeur **MSGREL(100,100)** indique qu'il n'y a eu aucun problème à court terme ou à long terme.
- La valeur **MSGREL(80,60)** indique que 20% des messages présentent actuellement des problèmes, mais qu'il s'agit également d'une amélioration par rapport à la valeur à long terme de 60.

Les clients peuvent continuer à transmettre et à recevoir du trafic multidiffusion même lorsque la connexion monodiffusion au gestionnaire de files d'attente est interrompue. Par conséquent, les données peuvent être obsolètes.

Fiabilité des messages de multidiffusion

Utilisez ces informations pour apprendre à définir l'abonnement à la multidiffusion WebSphere MQ et l'historique des messages.

La mise en mémoire tampon des données transmises par WebSphere MQ(historique des messages à conserver à l'extrémité émettrice de la liaison) est un élément clé pour surmonter l'échec de la transmission avec la multidiffusion. Ce processus signifie qu'aucune mise en mémoire tampon des messages n'est requise dans le processus d'application d'insertion car WebSphere MQ fournit la fiabilité. La taille de cet historique est configurée via l'objet d'informations de communication (COMMINFO), comme décrit dans les informations suivantes. Un tampon de transmission plus grand signifie qu'il y a plus d'historique de transmission à retransmettre si nécessaire, mais en raison de la nature de la multidiffusion, une distribution assurée à 100% ne peut pas être prise en charge.

L'historique des messages de multidiffusion WebSphere MQ est contrôlé dans l'objet d'informations de communication (COMMINFO) par l'attribut **MSGHIST** :

MSGHIST
Cette valeur correspond à la quantité d'historique des messages, en kilooctets, qui est conservée par le système pour gérer les retransmissions dans le cas des accusés de réception négatifs.

La valeur `0` indique le niveau de fiabilité le moins élevé. La valeur par défaut est `100` Ko.

L'historique des nouveaux abonnements de multidiffusion WebSphere MQ est contrôlé dans l'objet d'informations de communication (COMMINFO) par l'attribut **NSUBHIST** :

NSUBHIST
L'historique du nouvel abonné indique si un abonné rejoignant un flot de publication reçoit autant de données que disponible, ou reçoit uniquement les publications effectuées à partir du moment de l'abonnement.

AUCUN
La valeur `NONE` permet à l'émetteur de transmettre uniquement la publication effectuée à partir du moment de l'abonnement. `NONE` est la valeur utilisée par défaut.

TOUT
La valeur `ALL` permet à l'émetteur de retransmettre autant d'historique de la rubrique que cela est connu. Dans certaines circonstances, cette situation peut donner un comportement similaire à celui des publications conservées.

Remarque : L'utilisation de la valeur de `ALL` peut avoir un impact négatif sur les performances s'il existe un historique de sujet volumineux car tout l'historique de sujet est retransmis.

Référence associée
DEFINI COMMINFO
ALTER COMMINFO

Tâches de multidiffusion avancées

Utilisez ces informations pour en savoir plus sur les tâches d'administration de multidiffusion WebSphere MQ avancées, telles que la configuration des fichiers `.ini` et l'interopérabilité avec WebSphere MQ LLM.

Pour plus d'informations sur la sécurité dans une installation de multidiffusion, voir Sécurité de multidiffusion .

Pontage entre les domaines de publication / abonnement multidiffusion et non multidiffusion

Utilisez ces informations pour comprendre ce qui se passe lorsqu'un diffuseur de publications non multidiffusion est publié dans une rubrique WebSphere MQ activée pour la multidiffusion.

Si un diffuseur de publications non multidiffusion est publié dans une rubrique définie comme **MCAST** activée et **BRIDGE** activée, le gestionnaire de files d'attente transmet le message en mode multidiffusion directement aux abonnés qui peuvent être en mode écoute. Un diffuseur de publications multidiffusion ne peut pas publier dans des rubriques qui ne sont pas activées pour la multidiffusion.

Les rubriques existantes peuvent être activées en mode multidiffusion en définissant les paramètres **MCAST** et **COMMINFO** d'un objet de rubrique. Pour plus d'informations sur ces paramètres, voir Concepts de multidiffusion initiaux .

L'attribut **BRIDGE** de l'objet COMMINFO contrôle les publications des applications qui n'utilisent pas la multidiffusion. Si **BRIDGE** est défini sur ENABLED et que le paramètre **MCAST** de la rubrique est également défini sur ENABLED, les publications des applications qui n'utilisent pas la multidiffusion sont routées vers les applications qui le font. Pour plus d'informations sur le paramètre **BRIDGE** , voir DEFINE COMMINFO.

Configuration des fichiers .ini pour la multidiffusion

Utilisez ces informations pour comprendre les zones de multidiffusion WebSphere MQ dans les fichiers `.ini` .

Une configuration de multidiffusion WebSphere MQ supplémentaire peut être effectuée dans un fichier `ini` . Le fichier `ini` spécifique que vous devez utiliser dépend du type d'application:

- Client: configurez le fichier *MQ_DATA_PATH*`/mqclient.ini` .
- Gestionnaire de files d'attente: configurez le fichier *MQ_DATA_PATH*`/qmgrs/`*QMNAME*`/qm.ini` .

où *MQ_DATA_PATH* est l'emplacement du répertoire de données WebSphere MQ (`/var/mqm/mqclient.ini`) et *QMNAME* est le nom du gestionnaire de files d'attente auquel s'applique le fichier `.ini` .

Le fichier `.ini` contient des zones permettant d'affiner le comportement de WebSphere MQ Multicast:

```
Multicast:
   Protocol              = IP | UDP
   IPVersion             = IPV4 | IPV6 | ANY | BOTH
   LimitTransRate        = DISABLED | STATIC | DYNAMIC
   TransRateLimit        = 100000
   SocketTTL             = 1
   Batch                 = NO
   Loop                  = 1
   Interface             = <IPaddress>
   FeedbackMode          = ACK | NACK | WAIT1
   HeartbeatTimeout      = 20000
   HeartbeatInterval     = 2000
```

Protocole

UDP
Dans ce mode, les paquets sont envoyés à l'aide du protocole UDP. Les éléments de réseau ne peuvent pas fournir d'assistance dans la distribution multidiffusion comme ils le font en mode IP. Le format de paquet reste compatible avec PGM. Il s'agit de la valeur par défaut.

Adresse IP
Dans ce mode, l'émetteur envoie des paquets IP bruts. Les éléments de réseau avec prise en charge de PGM aident à la distribution de paquets multidiffusion fiable. Ce mode est entièrement compatible avec la norme PGM.

IPVersion

IPV4
Communiquez uniquement à l'aide du protocole IPv4 . Il s'agit de la valeur par défaut.

IPV6
Communiquez uniquement à l'aide du protocole IPv6 .

ANY
Communiquez à l'aide de IPv4, IPv6ou des deux, selon le protocole disponible.

LES DEUX
Prend en charge la communication à l'aide de IPv4 et IPv6.

LimitTransDébit

DESACTIVE
Il n'y a pas de contrôle du débit de transmission. Il s'agit de la valeur par défaut.

STATIQUE
Implémente le contrôle du débit de transmission statique. L'émetteur ne transmet pas à un débit supérieur au débit spécifié par le paramètre de limite TransRate.

DYNAMIQUE
L'émetteur adapte son débit de transmission en fonction de la rétroaction qu'il reçoit des récepteurs. Dans ce cas, la limite de débit de transmission ne peut pas être supérieure à la valeur spécifiée par le paramètre TransRateLimit. L'émetteur tente d'atteindre un débit de transmission optimal.

Limite TransRate
Limite de débit de transmission en kbit/s.

SocketTTL
La valeur de SocketTTL détermine si le trafic de multidiffusion peut passer par un routeur, ou le nombre de routeurs qu'il peut traverser.

Lot
Contrôle si les messages sont envoyés par lots ou immédiatement. Il existe 2 valeurs possibles:

- *NON* Les messages ne sont pas envoyés par lots, ils sont envoyés immédiatement.
- *OUI* Les messages sont traités par lots.

Boucle
Définissez la valeur sur 1 pour activer la boucle de multidiffusion. La boucle de multidiffusion définit si les données envoyées sont ou non rebouclées sur l'hôte.

utilisateur
Adresse IP de l'interface sur laquelle transite le trafic multidiffusion. Pour plus d'informations et pour résoudre les problèmes, voir: Test d'applications multidiffusion sur un réseau non multidiffusion et Définition du réseau approprié pour le trafic multidiffusion

FeedbackMode

NACK
Commentaires en retour par des accusés de réception négatifs. Il s'agit de la valeur par défaut.

Accusé de réception
Commentaires en retour par des accusés de réception positifs.

WAIT1
Rétroaction par des accusés de réception positifs où l'émetteur attend seulement 1 ACK de n'importe lequel des récepteurs.

HeartbeatTimeout
Délai d'attente du signal de présence en millisecondes. La valeur 0 indique que les événements de délai de signal de présence ne sont pas émis par le ou les récepteurs de la rubrique. La valeur par défaut est 20000.

HeartbeatInterval
Intervalle des pulsations en millisecondes. La valeur 0 indique qu'aucun signal de présence n'est envoyé. L'intervalle des pulsations doit être considérablement inférieur à la valeur **`HeartbeatTimeout`** pour éviter les faux événements de délai d'attente des pulsations. La valeur par défaut est 2000.

Interopérabilité multidiffusion avec WebSphere MQ Low Latency Messaging

Utilisez ces informations pour comprendre l'interopérabilité entre WebSphere MQ Multicast et WebSphere MQ Low Latency Messaging (LLM).

Le transfert de contenu de base est possible pour une application utilisant LLM, avec une autre application utilisant la multidiffusion pour échanger des messages dans les deux sens. Bien que la multidiffusion utilise la technologie LLM, le produit LLM lui-même n'est pas intégré. Par conséquent, il est possible d'installer LLM et WebSphere MQ Multicast, et d'utiliser et de traiter les deux produits séparément.

Les applications LLM qui communiquent avec la multidiffusion peuvent avoir besoin d'envoyer et de recevoir des propriétés de message. Les propriétés de message WebSphere MQ et les zones MQMD sont transmises en tant que propriétés de message LLM avec des codes de propriété de message LLM spécifiques, comme indiqué dans le tableau suivant:

Tableau 10. Propriétés de message WebSphere MQ vers les mappages de propriété LLM WebSphere MQ			
Propriété WebSphere MQ	**WebSphere MQ Type de propriété LLM**	**Type de propriété LLM**	**Code de propriété LLM**
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_Chaîne	-1007
MQMD.Priority	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1008
MQMD.Persistence	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1009
MQMD.MsgId	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_ByteArray	-1010
MQMD.BackoutCount	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1012
MQMD.ReplyToQ	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_Chaîne	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_Chaîne	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_Chaîne	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_Chaîne	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_Chaîne	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

Pour plus d'informations sur LLM, voir la documentation du produit LLM: WebSphere MQ Low Latency Messaging.

Administration d'HP Integrity NonStop Server

Utilisez ces informations pour en savoir plus sur les tâches d'administration du client IBM WebSphere MQ pour HP Integrity NonStop Server.

Deux tâches d'administration sont disponibles:

1. Démarrage manuel de TMF/Gateway à partir de Pathway.
2. Arrêt de TMF/Gateway à partir de Pathway.

Démarrage manuel de TMF/Gateway à partir de Pathway

Vous pouvez autoriser Pathway à démarrer automatiquement TMF/Gateway lors de la première demande d'inscription, ou vous pouvez démarrer manuellement TMF/Gateway à partir de Pathway.

Procédure

Pour démarrer manuellement TMF/Gateway à partir de Pathway, entrez la commande PATHCOM suivante:

```
START SERVER <server_class_name>
```

Si une application client effectue une demande d'inscription avant que TMF/Gateway ne termine la récupération des transactions en attente de validation, la demande est suspendue jusqu'à 1 seconde. Si

la récupération ne se termine pas dans ce délai, l'inscription est rejetée. Le client reçoit ensuite une erreur MQRC_UOW_ENLISTMENT_ERROR suite à l'utilisation d'un MQI transactionnel.

Arrêt de TMF/Gateway à partir de Pathway

Cette tâche explique comment arrêter TMF/Gateway à partir de Pathway et comment redémarrer TMF/Gateway après l'avoir arrêté.

Procédure

1. Pour empêcher toute nouvelle demande d'inscription à TMF/Gateway, entrez la commande suivante:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. Pour que TMF/Gateway puisse effectuer des opérations en cours et se terminer, entrez la commande suivante:

   ```
   STOP SERVER <server_class_name>
   ```

3. Pour permettre à TMF/Gateway de redémarrer automatiquement lors de la première inscription ou manuellement, en suivant les étapes 1 et 2, entrez la commande suivante:

   ```
   THAW SERVER <server_class_name>
   ```

   Les applications ne peuvent pas effectuer de nouvelles demandes d'inscription et il n'est pas possible d'émettre la commande **START** tant que vous n'émettez pas la commande **THAW** .

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 le fonctionnement de tout produit, programme ou service non expressément référencés par IBM.

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

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

Pour 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 :

Licence de propriété intellectuelle Droit juridique et droit de la propriété intellectuelle 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 Software Interoperability Coordinator, Département 49XA 3605 Highway 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 relatives aux produits non IBM ont été obtenues via les fournisseurs de ces produits, leurs annonces publiées ou d'autres sources publiquement 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 IBM 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: