7.5

IBM WebSphere MQ verwalten

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen” auf Seite 171 gelesen werden.

Diese Ausgabe bezieht sich auf Version 7 Release 5 von IBM® WebSphere MQ und auf alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Wenn Sie Informationen an IBMsenden, erteilen Sie IBM ein nicht ausschließliches Recht, die Informationen in beliebiger Weise zu verwenden oder zu verteilen, ohne dass eine Verpflichtung für Sie entsteht.

Inhaltsverzeichnis

IBM WebSphere MQ verwalten

Die Verwaltung von Warteschlangenmanagern und zugeordneten Ressourcen umfasst die Tasks, die Sie häufig ausführen, um diese Ressourcen zu aktivieren und zu verwalten. Wählen Sie die Methode aus, die Sie bevorzugen, um die Warteschlangenmanager und die zugehörigen Ressourcen zu verwalten.

Sie können IBM WebSphere MQ-Objekte lokal oder fern verwalten (siehe „Lokale und ferne Verwaltung” auf Seite 7).

Es gibt eine Reihe unterschiedlicher Methoden, die Sie zum Erstellen und Verwalten Ihrer Warteschlangenmanager und ihrer zugehörigen Ressourcen in IBM WebSphere MQ verwenden können. Zu diesen Methoden gehören Befehlszeilenschnittstellen, eine grafische Benutzerschnittstelle und eine Verwaltungs-API. Weitere Informationen zu diesen einzelnen Schnittstellen finden Sie in den Abschnitten und Links in diesem Thema.

Es gibt verschiedene Gruppen von Befehlen, die Sie abhängig von der Plattform für die Verwaltung von IBM WebSphere MQ verwenden können:

- „IBM WebSphere MQ-Steuerbefehle” auf Seite 5
- „IBM WebSphere MQ Scriptbefehle (MQSC)” auf Seite 5
- „Programmierbare Befehlsformate (PCFs)” auf Seite 6

Es gibt auch die folgenden anderen Optionen zum Erstellen und Verwalten von IBM WebSphere MQ-Objekten:

- „IBM WebSphere MQ Explorer” auf Seite 6
- „Windows Default Configuration Application (Anwendung für Standardkonfiguration).” auf Seite 7
- „Microsoft Cluster Service (MSCS)” auf Seite 7

Sie können einige Verwaltungs-und Überwachungstasks sowohl für lokale als auch für ferne Warteschlangenmanager mithilfe von PCF-Befehlen automatisieren. Diese Befehle können auch durch die Verwendung der IBM WebSphere MQ-Verwaltungsschnittstelle (MQAI) auf einigen Plattformen vereinfacht werden. Weitere Informationen zum Automatisieren von Verwaltungstasks finden Sie im Abschnitt „Verwaltungstasks automatisieren” auf Seite 8.

IBM WebSphere MQ-Steuerbefehle

Mithilfe von Steuerbefehlen können Sie Verwaltungstasks auf den Warteschlangenmanagern selbst ausführen.

IBM WebSphere MQ für Windows UNIX and Linux® -Systeme stellt die *Steuerbefehle* bereit, die Sie in der Systembefehlszeile absetzen.

Die Steuerbefehle werden im Abschnitt Warteschlangenmanager erstellen und verwaltenbeschrieben. Die Befehlsreferenz für die Steuerbefehle finden Sie in IBM WebSphere MQ-Steuerbefehle.

IBM WebSphere MQ Scriptbefehle (MQSC)

Verwenden Sie MQSC-Befehle zum Verwalten von Warteschlangenmanagerobjekten, einschließlich des Warteschlangenmanagers selbst, Warteschlangen, Prozessdefinitionen, Namenslisten, Kanäle, Clientverbindungskanäle, Empfangsprogramme, Services und Authentifizierungsinformationsobjekte.

MQSC-Befehle werden mit dem Befehl `runmqsc` an einen Warteschlangenmanager ausgegeben. Sie können dies interaktiv ausführen, Befehle über eine Tastatur ausgeben oder die Standardeingabeeinheit (stdin) umleiten, um eine Folge von Befehlen aus einer ASCII-Textdatei auszuführen. In beiden Fällen ist das Format der Befehle identisch.

Sie können den Befehl `runmqsc` in drei Modi ausführen, je nachdem, welche Flags im Befehl festgelegt wurden:

- *Prüfmodus* , bei dem die MQSC-Befehle in einem lokalen Warteschlangenmanager geprüft werden, aber nicht ausgeführt werden
- *Direktmodus* : Die MQSC-Befehle werden auf einem lokalen Warteschlangenmanager ausgeführt.
- *Indirekter Modus* , in dem die MQSC-Befehle auf einem fernen Warteschlangenmanager ausgeführt werden

Objektattribute, die in MQSC-Befehlen angegeben werden, werden in diesem Abschnitt in Großbuchstaben (z. B. RQMNAME) angezeigt, obwohl die Groß-/Kleinschreibung nicht beachtet werden muss. Die Namen von MQSC-Befehlsattributen sind auf acht Zeichen begrenzt.

MQSC-Befehle sind auf allen Plattformen verfügbar. MQSC-Befehle werden in Befehlssätzen verglichen zusammengefasst.

Unter Windows, UNIX oder Linux können Sie die MQSC-Befehle als einzelne Befehle über die Systembefehlszeile ausgeben. Zur Ausgabe komplizierter Befehle oder mehrerer Befehle können Sie MQSC-Befehle in einer Datei zusammenfassen, die Sie über die Systembefehlszeile von Windows, UNIX oder Linux ausführen. Die WebSphere MQ-Scriptbefehle können an einen fernen Warteschlangenmanager gesendet werden. Ausführliche Informationen enthält die MQSC-Referenz.

„Scriptbefehle (MQSC)” auf Seite 76 enthält eine Beschreibung der einzelnen MQSC-Befehle und ihre Syntax.

Weitere Informationen zur Verwendung von MQSC-Befehlen für die lokale Verwaltung finden Sie unter „Lokale Verwaltungsaufgaben mithilfe von MQSC-Befehlen ausführen” auf Seite 75.

Programmierbare Befehlsformate (PCFs)

Programmierbare Befehlsformate (PCFs) definieren Befehls-und Antwortnachrichten, die zwischen einem Programm und einem beliebigen WS-Manager (der PCFs unterstützt) in einem Netz ausgetauscht werden können. Sie können PCF-Befehle in einem Systemverwaltungsanwendungsprogramm für die Verwaltung von IBM WebSphere MQ-Objekten verwenden: Authentifizierungsinformationsobjekte, Kanäle, Kanallistener, Namenslisten, Prozessdefinitionen, Warteschlangenmanager, Warteschlangen, Services und Speicherklassen. Die Anwendung kann von einem einzigen Punkt im Netz aus ausgeführt werden, um die Befehls-und Antwortinformationen mit jedem WS-Manager, lokal oder fern mit dem lokalen WS-Manager zu kommunizieren.

Weitere Informationen zu PCFs finden Sie im Abschnitt „Einführung in Programmierbare Befehlsformate” auf Seite 9.

Informationen zur Definition von PCFs und Strukturen für die Befehle und Antworten finden Sie im Abschnitt Programmierbare Befehlsformate-Referenz .

IBM WebSphere MQ Explorer

Mit IBM WebSphere MQ Explorerkönnen Sie die folgenden Aktionen ausführen:

- Definieren und Steuern verschiedener Ressourcen, einschließlich Warteschlangenmanager, Warteschlangen, Prozessdefinitionen, Namenslisten, Kanäle, Clientverbindungskanäle, Empfangsprogramme, Services und Cluster.
- Starten oder Stoppen eines lokalen Warteschlangenmanagers und der zugehörigen Prozesse.
- Zeigen Sie die Warteschlangenmanager und die zugehörigen Objekte auf Ihrer Workstation oder von anderen Workstations an.
- Überprüfen Sie den Status von Warteschlangenmanagern, Clustern und Kanälen.
- Überprüfen Sie, welche Anwendungen, Benutzer oder Kanäle eine bestimmte Warteschlange geöffnet haben, aus dem Warteschlangenstatus.

Auf Windows -und Linux -Systemen können Sie IBM WebSphere MQ Explorer über das Systemmenü, die ausführbare Datei `MQExplorer` oder den Befehl **strmqcfg** starten.

Unter Linuxmüssen Sie zum erfolgreichen Starten von IBM WebSphere MQ Explorer in der Lage sein, eine Datei in Ihr Ausgangsverzeichnis zu schreiben, und das Ausgangsverzeichnis muss vorhanden sein.

Sie können IBM WebSphere MQ Explorer zum Verwalten von fernen Warteschlangenmanagern auf anderen Plattformen, einschließlich z/OS, verwenden, um Details anzuzeigen und das SupportPac MS0T herunterzuladen. Weitere Informationen finden Sie im Abschnitt https://www.ibm.com/support/docview.wss?uid=swg24021041.

Weitere Informationen finden Sie unter „Verwaltung mit IBM WebSphere MQ Explorer" auf Seite 58.

Windows Default Configuration Application (Anwendung für Standardkonfiguration).

Sie können das Standardkonfigurationsprogramm von Windows verwenden, um eine *Startergruppe* (oder Standardgruppe) von IBM WebSphere MQ -Objekten zu erstellen. Eine Zusammenfassung der erstellten Standardobjekte finden Sie in Tabelle 1: Objekte, die von der Windows -Standardkonfigurationsanwendung erstellt wurden.

Microsoft Cluster Service (MSCS)

Microsoft Cluster Service (MSCS) ermöglicht es Ihnen, Server in einem *Cluster*zu verbinden, was eine höhere Verfügbarkeit von Daten und Anwendungen ermöglicht und die Verwaltung des Systems vereinfacht. MSCS kann Server-oder Anwendungsfehler automatisch erkennen und wiederherstellen.

Es ist wichtig, Cluster im MSCS-Sinn nicht mit IBM WebSphere MQ-Clustern zu verwechseln. Sie unterscheiden sich wie folgt:

IBM WebSphere MQ-Cluster
Gruppen von mindestens zwei Warteschlangenmanagern auf mindestens einem Computer, die automatisch miteinander verbunden werden und Warteschlangen zum Zweck des Lastausgleichs und der Redundanz gemeinsam nutzen.

MSCS-Cluster
Gruppen von Computern, die miteinander verbunden und so konfiguriert sind, dass MSCS bei einem Fehler auf einem der Computer eine *Überbrückung* ausführt, wobei die Statusdaten von Anwendungen von dem fehlerhaften Computer auf einen anderen Computer im Cluster übertragen und die Anwendungen dort erneut gestartet werden.

Unterstützung von Microsoft Cluster Service (MSCS) enthält detaillierte Informationen zur Konfiguration Ihres IBM WebSphere MQ for Windows -Systems für die Verwendung von MSCS.

Zugehörige Konzepte
WebSphere MQ - Technische Übersicht
„Lokale IBM WebSphere MQ -Objekte verwalten" auf Seite 71
In diesem Abschnitt erfahren Sie, wie Sie lokale IBM WebSphere MQ-Objekte verwalten können, um Anwendungsprogramme zu unterstützen, die die Schnittstelle für Nachrichtenwarteschlangen (Message Queue Interface, MQI) verwenden. In diesem Kontext bedeutet lokale Verwaltung das Erstellen, Anzeigen, Ändern, Kopieren und Löschen von IBM WebSphere MQ-Objekten.

„Ferne IBM WebSphere MQ -Objekte verwalten" auf Seite 108

Überlegungen zur Vorgehensweise bei Verlust des Kontakts zum XA-Ressourcenmanager
Zugehörige Tasks
Planung
Konfiguration
Zugehörige Verweise
Transaktionsunterstützungsszenarios

Lokale und ferne Verwaltung

Sie können WebSphere MQ -Objekte lokal oder fern verwalten.

Lokale Verwaltung bedeutet, dass Verwaltungsaufgaben für alle Warteschlangenmanager, die Sie definiert haben, auf dem lokalen System ausgeführt werden. Sie können auf andere Systeme zugreifen, z. B. über

das TCP/IP-Terminal-Emulationsprogramm **telnet** , und führen die Verwaltung dort aus. In WebSphere MQkönnen Sie dies als lokale Verwaltung betrachten, da keine Kanäle beteiligt sind, d. h., die Kommunikation wird vom Betriebssystem verwaltet.

WebSphere MQ unterstützt die Verwaltung von einem zentralen Ansprechpartner über die so genannte *Fernverwaltung*. Auf diese Weise können Sie Befehle von Ihrem lokalen System absetzen, die auf einem anderen System verarbeitet werden und auch für WebSphere MQ Explorer gelten. Sie können beispielsweise einen fernen Befehl absetzen, um eine Warteschlangendefinition in einem fernen Warteschlangenmanager zu ändern. Sie müssen sich nicht bei diesem System anmelden, obwohl Sie die entsprechenden Kanäle definiert haben müssen. Der Warteschlangenmanager und der Befehlsserver auf dem Zielsystem müssen aktiv sein.

Einige Befehle können auf diese Weise nicht abgesetzt werden, insbesondere die Erstellung oder den Start von Warteschlangenmanagern und den Start von Befehlsservern. Um diese Art von Verwaltungsaufgaben auszuführen, müssen Sie sich entweder am fernen System anmelden und die Befehle dort ausgeben oder Sie müssen einen Prozess erstellen, der die Befehle automatisch ausführt. Diese Einschränkung gilt auch für WebSphere MQ Explorer.

Eine ausführlichere Beschreibung der Fernverwaltung finden Sie im Abschnitt „Ferne IBM WebSphere MQ -Objekte verwalten” auf Seite 108.

Verwendung von IBM WebSphere MQ -Steuerbefehlen

In diesem Abschnitt wird die Verwendung der IBM WebSphere MQ-Steuerbefehle beschrieben.

Wenn Sie Steuerbefehle wie zum Beispiel ausgeben möchten, muss Ihre Benutzer-ID zur Gruppe 'mqm' gehören. Weitere Informationen hierzu finden Sie im Abschnitt Berechtigung zur Verwaltung von WebSphere MQ auf UNIX-, Linux -und Windows -Systemen . Beachten Sie zusätzlich die folgenden umgebungsspezifischen Informationen:

WebSphere MQ für Windows
Alle Steuerbefehle können über die Befehlszeile ausgegeben werden. Bei Befehlsnamen und ihren Flags muss die Groß-/Kleinschreibung nicht beachtet werden. Sie können sie also in Großbuchstaben, in Kleinbuchstaben oder kombiniert eingeben. Bei Argumenten von Steuerbefehlen (z. B. die Namen der Warteschlange) hingegen muss die Groß-/Kleinschreibung beachtet werden.

In den Syntaxbeschreibungen dient der Bindestrich (-) als Markierungsanzeiger. Statt des Bindestrichs können Sie auch den Schrägstrich (/) verwenden.

WebSphere MQ for UNIX and Linux-Systeme
Alle WebSphere MQ -Steuerbefehle können über eine Shell ausgegeben werden. Bei allen Befehlen muss die Groß-/Kleinschreibung beachtet werden.

Ein Teil der Steuerbefehle kann mit dem IBM WebSphere MQ ausgegeben werden.

Weitere Informationen hierzu finden Sie im Abschnitt WebSphere MQ -Steuerbefehle .

Verwaltungstasks automatisieren

Sie können entscheiden, dass es für Ihre Installation von Vorteil wäre, einige Verwaltungs- und Überwachungstasks zu automatisieren. Sie können Verwaltungstasks sowohl für lokale als auch für ferne Warteschlangenmanager mit Hilfe von PCF-Befehlen (PCF = Programmable Command Format) automatisieren. In diesem Abschnitt wird vorausgesetzt, dass Sie Erfahrung mit der Verwaltung von WebSphere MQ -Objekten haben.

PCF-Befehle

WebSphere MQ PCF-Befehle (Programmable Command Format) können verwendet werden, um Verwaltungstasks in einem Verwaltungsprogramm zu programmieren. Auf diese Weise können Sie von einem Programm aus Warteschlangenmanagerobjekte (Warteschlangen, Prozessdefinitionen, Namenslisten, Kanäle, Clientverbindungskanäle, Empfangsprogramme, Services und Authentifizierungsinformationsobjekte) bearbeiten und sogar die Warteschlangenmanager selbst bearbeiten.

PCF-Befehle decken den gleichen Funktionsumfang ab, der von MQSC-Befehlen bereitgestellt wird. Sie können ein Programm schreiben, um PCF-Befehle an jeden WS-Manager im Netz von einem einzelnen Knoten aus auszugeben. Auf diese Weise können Sie Verwaltungsaufgaben zentralisieren und automatisieren.

Jeder PCF-Befehl ist eine Datenstruktur, die in den Anwendungsdatenteil einer WebSphere MQ -Nachricht eingebettet ist. Jeder Befehl wird mit der MQI-Funktion MQPUT auf dieselbe Weise wie jede andere Nachricht an den Zielwarteschlangenmanager gesendet. Wenn der Befehlsserver auf dem Warteschlangenmanager ausgeführt wird, der die Nachricht empfängt, interpretiert der Befehlsserver die Nachricht als Befehlsnachricht und führt den Befehl aus. Zum Abrufen der Antworten gibt die Anwendung einen MQGET -Aufruf aus und die Antwortdaten werden in einer anderen Datenstruktur zurückgegeben. Die Anwendung kann dann die Antwort verarbeiten und entsprechend handeln.

Anmerkung: Im Gegensatz zu MQSC-Befehlen befinden sich PCF-Befehle und ihre Antworten nicht in einem Textformat, das Sie lesen können.

Kurz, dies sind einige der Dinge, die zum Erstellen einer PCF-Befehlsnachricht erforderlich sind:

Nachrichtendeskriptor
Dies ist ein WebSphere MQ -Standardnachrichtendeskriptor, in dem Folgendes gilt:

- Der Nachrichtentyp (*MsqType*) ist MQMT_REQUEST.
- Nachrichtenformat (*Format*) ist MQFMT_ADMIN.

Anwendungsdaten
Enthält die PCF-Nachricht (einschließlich des PCF-Headers), in der Folgendes gilt:

- Der PCF-Nachrichtentyp (*Type*) gibt MQCFT_COMMAND an.
- Die Befehls-ID gibt den Befehl an, beispielsweise *Change Queue* (MQCMD_CHANGE_Q).

Eine vollständige Beschreibung der PCF-Datenstrukturen und deren Implementierung finden Sie in „Einführung in Programmierbare Befehlsformate” auf Seite 9.

PCF-Objektattribute

Objektattribute in PCF sind nicht auf acht Zeichen beschränkt, da sie für MQSC-Befehle verwendet werden. Sie werden in diesem Handbuch in Kursivschrift angezeigt. Das PCF-Äquivalent von RQMNAME ist beispielsweise *RemoteQMgrName*.

Escape-PCFs

Escape-PCFs sind PCF-Befehle, die MQSC-Befehle innerhalb des Nachrichtentexts enthalten. Sie können PCFs verwenden, um Befehle an einen fernen Warteschlangenmanager zu senden. Weitere Informationen zu Escape-PCFs finden Sie unter Escape .

Einführung in Programmierbare Befehlsformate

Programmierbare Befehlsformate (PCFs) definieren Befehls-und Antwortnachrichten, die zwischen einem Programm und einem beliebigen WS-Manager (der PCFs unterstützt) in einem Netz ausgetauscht werden können. PCFs vereinfachen die Verwaltung von Warteschlangenmanagern und andere Netzadministrationssysteme. Sie können eingesetzt werden, um das Problem der komplexen Verwaltung verteilter Netzwerke zu lösen, insbesondere, wenn Netzwerke in Größe und Komplexität wachsen.

Die in diesem Produkt beschriebenen programmierbaren Befehlsformate werden von folgenden Systemen unterstützt:

- IBM WebSphere MQ für AIX
- IBM WebSphere MQ für HP-UX
- IBM WebSphere MQ für Linux
- IBM WebSphere MQ für Solaris
- IBM WebSphere MQ für Windows

- IBM WebSphere MQ for HP Integrity NonStop Server

Probleme mit PCF-Befehlen lösen

Die Verwaltung verteilter Netze kann komplex werden. Die Probleme der Administration wachsen weiter, da die Netzwerke in Größe und Komplexität zunehmen.

Beispiele für die Verwaltung von Messaging und Warteschlangensteuerung sind:

- Ressourcenmanagement.

  Zum Beispiel die Erstellung und Löschung von Warteschlangen.
- Leistungsüberwachung.

  Beispiel: Maximale Warteschlangenlänge oder Nachrichtenrate.
- Steuerung.

  Beispiel für die Optimierung von Warteschlangenparametern wie der maximalen Warteschlangenlänge, der maximalen Nachrichtenlänge und der Aktivierung und Inaktivierung von Warteschlangen.
- Nachrichtenweiterleitung.

  Definition von alternativen Routen durch ein Netzwerk.

WebSphere MQ -PCF-Befehle können verwendet werden, um die Verwaltung von Warteschlangenmanagern und anderen Netzen zu vereinfachen. Mithilfe von PCF-Befehlen können Sie eine einzelne Anwendung verwenden, um die Netzverwaltung über einen einzelnen Warteschlangenmanager im Netz auszuführen.

Was sind PCFs?

PCFs definieren Befehls-und Antwortnachrichten, die zwischen einem Programm und einem beliebigen Warteschlangenmanager (der PCFs unterstützt) in einem Netz ausgetauscht werden können. Sie können PCF-Befehle in einem Systemmanagementanwendungsprogramm für die Verwaltung von WebSphere MQ -Objekten verwenden: Authentifizierungsinformationsobjekte, Kanäle, Kanallistener, Namenslisten, Prozessdefinitionen, Warteschlangenmanager, Warteschlangen, Services und Speicherklassen. Die Anwendung kann von einem einzigen Punkt im Netz aus ausgeführt werden, um die Befehls-und Antwortinformationen mit jedem WS-Manager, lokal oder fern mit dem lokalen WS-Manager zu kommunizieren.

Jeder Warteschlangenmanager verfügt über eine Verwaltungswarteschlange mit einem Standardwarteschlangennamen, und Ihre Anwendung kann PCF-Befehlsnachrichten an diese Warteschlange senden. Jeder Warteschlangenmanager verfügt außerdem über einen Befehlsserver, um die Befehlsnachrichten aus der Verwaltungswarteschlange zu bedienen. PCF-Befehlsnachrichten können daher von jedem Warteschlangenmanager im Netz verarbeitet werden, und die Antwortdaten können mit Hilfe der angegebenen Antwortwarteschlange an Ihre Anwendung zurückgegeben werden. PCF-Befehle und Antwortnachrichten werden über die normale Nachrichtenwarteschlangenschnittstelle (MQI) gesendet und empfangen.

Eine Liste der verfügbaren PCF-Befehle, einschließlich ihrer Parameter, finden Sie im Abschnitt Definitionen der programmierbaren Befehlsformate .

Programmierbare Befehlsformate verwenden

Sie können PCFs in einem Systemmanagementprogramm für die WebSphere MQ -Fernverwaltung verwenden.

Dieser Abschnitt enthält:

- „PCF-Befehlsnachrichten” auf Seite 11
- „Antworten” auf Seite 13
- Regeln für die Benennung von IBM WebSphere MQ -Objekten
- „Berechtigungsprüfung für PCF-Befehle” auf Seite 15

PCF-Befehlsnachrichten

PCF-Befehlsnachrichten bestehen aus einem PCF-Header, den in diesem Header angegebenen Parametern und auch benutzerdefinierte Nachrichtendaten. Die Nachrichten werden unter Verwendung von Nachrichtenwarteschlangenschnittstellenaufrufen ausgegeben.

Jeder Befehl und seine Parameter werden als separate Befehlsnachricht gesendet, die einen PCF-Header gefolgt von einer Reihe von Parameterstrukturen enthält. Einzelheiten zum PCF-Header finden Sie im Abschnitt MQCFH-PCF-Header und in einem Beispiel für eine Parameterstruktur. Weitere Informationen finden Sie im Abschnitt MQCFST-PCF-Zeichenfolgeparameter . Der PCF-Header gibt den Befehl und die Anzahl der Parameterstrukturen an, die in derselben Nachricht folgen. Jede Parameterstruktur stellt einen Parameter für den Befehl bereit.

Antworten auf die Befehle, die vom Befehlsserver generiert werden, haben eine ähnliche Struktur. Es gibt einen PCF-Header, gefolgt von einer Reihe von Parameterstrukturen. Antworten können aus mehr als einer Nachricht bestehen, aber Befehle bestehen immer nur aus einer Nachricht.

Auf anderen Plattformen als z/OSwird die Warteschlange, an die die PCF-Befehle gesendet werden, immer als SYSTEM.ADMIN.COMMAND.QUEUE.

Informationen zum Absetzen von PCF-Befehlsnachrichten

Verwenden Sie die normalen MQI-Aufrufe (MQI = Message Queue Interface), MQPUT, MQGET usw. zum Einreihen und Abrufen von PCF-Befehls-und Antwortnachrichten in und aus ihren Warteschlangen.

Anmerkung:

Stellen Sie sicher, dass der Befehlsserver auf dem Zielwarteschlangenmanager ausgeführt wird, damit der PCF-Befehl in diesem WS-Manager verarbeitet wird.

Eine Liste der bereitgestellten Headerdateien finden Sie unter WebSphere MQ COPY-, Header-, Include- und Moduldateien.

Nachrichtendeskriptor für einen PCF-Befehl

Der Nachrichtendeskriptor WebSphere MQ ist vollständig in MQMD-Nachrichtendeskriptordokumentiert.

Eine PCF-Befehlsnachricht enthält die folgenden Felder im Nachrichtendeskriptor:

Report
Jeder gültige Wert, wie erforderlich.

MsgType
Dieses Feld muss MQMT_REQUEST sein, um eine Nachricht anzugeben, die eine Antwort erfordert.

Expiry
Jeder gültige Wert, wie erforderlich.

Feedback
Auf 'MQFB_NONE' gesetzt

Encoding
Wenn Sie an eines der folgenden Systeme senden, setzen Sie dieses Feld auf die für die Nachrichtendaten verwendete Codierung: Windows, UNIX oder LinuxFalls erforderlich, wird die Konvertierung durchgeführt.

CodedCharSetId
Beim Senden an Systeme unterWindows-, UNIX -oder Linux -Systeme setzen dieses Feld auf die ID des codierten Zeichensatzes, die für die Nachrichtendaten verwendet wird. Bei Bedarf wird die Konvertierung durchgeführt.

Format
Setzen Sie sie auf MQFMT_ADMIN.

Priority
Jeder gültige Wert, wie erforderlich.

Persistence
Jeder gültige Wert, wie erforderlich.

MsgId
Die sendende Anwendung kann jeden beliebigen Wert angeben, oder MQMI_NONE kann angegeben werden, um den Warteschlangenmanager anzufordern, eine eindeutige Nachrichten-ID zu generieren.

CorrelId
Die sendende Anwendung kann einen beliebigen Wert angeben, oder MQCI_NONE kann angegeben werden, um keine Korrelations-ID anzugeben.

ReplyToQ
Der Name der Warteschlange, die die Antwort empfangen soll.

ReplyToQMgr
Der Name des Warteschlangenmanagers für die Antwort (oder leer).

Nachrichtenkontextfelder
Diese Felder können, wie erforderlich, auf alle gültigen Werte gesetzt werden. In der Regel wird die Put-Nachrichtenoption MQPMO_DEFAULT_CONTEXT verwendet, um die Nachrichtenkontextfelder auf die Standardwerte zu setzen.

Wenn Sie eine MQMD-Struktur der Version 2 verwenden, müssen Sie die folgenden zusätzlichen Felder festlegen:

GroupId
Auf MQGI_NONE gesetzt

MsgSeqNumber
Auf 1 setzen

Offset
Auf 0 setzen

MsgFlags
Auf MQMF_NONE gesetzt

OriginalLength
Auf MQOL_UNDEFINED setzen

Benutzerdaten senden

Die PCF-Strukturen können auch zum Senden benutzerdefinierter Nachrichtendaten verwendet werden. In diesem Fall muss das Nachrichtendeskriptorfeld *Format* auf MQFMT_PCF gesetzt werden.

PCF-Nachrichten in einer angegebenen Warteschlange senden und empfangen

PCF-Nachrichten an eine angegebene Warteschlange senden

Um eine Nachricht an eine angegebene Warteschlange zu senden, konvertiert der mqPutBag-Aufruf den Inhalt des angegebenen Sacks in eine PCF-Nachricht und sendet die Nachricht an die angegebene Warteschlange. Der Inhalt des Beutels wird nach dem Anruf unverändert gelassen.

Als Eingabe für diesen Aufruf müssen Sie Folgendes angeben:

- Eine MQI-Verbindungskennung.
- Eine Objektkennung für die Warteschlange, in die die Nachricht gestellt werden soll.
- Ein Nachrichtendeskriptor. Weitere Informationen zum Nachrichtendeskriptor finden Sie im Abschnitt MQMD-Nachrichtendeskriptor .
- Einreihen von Nachrichtenoptionen mit Hilfe der MQPMO-Struktur. Weitere Informationen zur MQPMO-Struktur finden Sie unter MQPMO-Put-message-Optionen .
- Der Handle der Tasche, die in eine Nachricht umgewandelt werden soll.

Anmerkung: Wenn die Tasche eine Verwaltungsnachricht enthält und der Aufruf 'mqAddInquiry' zum Einfügen von Werten in die Tasche verwendet wurde, muss der Wert des Datenelements 'MQIASY_COMMAND' ein Befehl INQUIRE sein, der von der MQAI erkannt wird.

Eine vollständige Beschreibung des mqPutBag-Aufrufs finden Sie unter mqPutBag .

PCF-Nachrichten aus einer angegebenen Warteschlange empfangen

Um eine Nachricht aus einer angegebenen Warteschlange zu empfangen, ruft der mqGetBag-Aufruf eine PCF-Nachricht aus einer angegebenen Warteschlange ab und konvertiert die Nachrichtendaten in einen Datenbehälter.

Als Eingabe für diesen Aufruf müssen Sie Folgendes angeben:

- Eine MQI-Verbindungskennung.
- Eine Objektkennung der Warteschlange, aus der die Nachricht gelesen werden soll.
- Ein Nachrichtendeskriptor. Innerhalb der MQMD-Struktur muss der Parameter `Format` MQFMT_ADMIN, MQFMT_EVENT oder MQFMT_PCF sein.

  Anmerkung: Wenn die Nachricht innerhalb einer UOWs (d. a. mit der Option MQGMO_SYNCPOINT) empfangen wird und die Nachricht ein nicht unterstütztes Format hat, kann die Arbeitseinheit zurückgesetzt werden. Die Nachricht wird dann in der Warteschlange wieder in die Warteschlange gestellt und kann mit dem MQGET-Aufruf anstelle des mqGetBag-Aufrufs abgerufen werden. Weitere Informationen zum Nachrichtendeskriptor finden Sie unter MQGMO-Get-message options .
- Abrufen von Nachrichtenoptionen unter Verwendung der MQGMO-Struktur. Weitere Informationen zur MQGMO-Struktur finden Sie im Abschnitt MQMD-Nachrichtendeskriptor .
- Der Handle der Tasche, die die konvertierte Nachricht enthalten soll.

Eine vollständige Beschreibung des mqGetBag-Aufrufs finden Sie in mqGetBag .

Antworten

Als Antwort auf jeden Befehl generiert der Befehlsserver eine oder mehrere Antwortnachrichten. Eine Antwortnachricht hat ein ähnliches Format wie eine Befehlsnachricht.

Der PCF-Header hat denselben Befehlskennungswert wie der Befehl, auf den es sich um eine Antwort handelt (Details hierzu finden Sie im Abschnitt MQCFH-PCF-Header). Die Nachrichten-ID und die Korrelations-ID werden entsprechend den Berichtsoptionen der Anforderung festgelegt.

Wenn der PCF-Header-Typ der Befehlsnachricht MQCFT_COMMAND ist, werden nur Standardantworten generiert. Solche Befehle werden auf allen Plattformen außer z/OSunterstützt. Ältere Anwendungen unterstützen PCF unter z/OSnicht. WebSphere MQ Windows Explorer ist eine solche Anwendung (IBM WebSphere MQ Explorer Version 6.0 oder höher unterstützt jedoch PCF unter z/OS).

Wenn der PCF-Header-Typ der Befehlsnachricht MQCFT_COMMAND_XR lautet, werden entweder erweiterte oder Standardantworten generiert. Solche Befehle werden unter z/OS und einigen anderen Plattformen unterstützt. Befehle, die unter z/OS ausgegeben werden, generieren nur erweiterte Antworten. Auf anderen Plattformen kann entweder der Typ der Antwort generiert werden.

Wenn ein einzelner Befehl einen generischen Objektnamen angibt, wird eine separate Antwort in einer eigenen Nachricht für jedes übereinstimmende Objekt zurückgegeben. Bei der Antwortgenerierung wird ein einzelner Befehl mit einem generischen Namen als mehrere Einzelbefehle behandelt (außer dem Steuerfeld MQCFC_LAST oder MQCFC_NOT_LAST). Andernfalls generiert eine Befehlsnachricht eine Antwortnachricht.

Bestimmte PCF-Antworten geben möglicherweise eine Struktur zurück, selbst wenn sie nicht angefordert wird. Diese Struktur wird in der Definition der Antwort (Definitionen der programmierbaren Befehlsformate) als *immer zurückgegeben*angezeigt. Der Grund, warum es für diese Antworten erforderlich ist, die Objekte in der Antwort zu benennen, um zu identifizieren, für welches Objekt die Daten gelten.

Nachrichtendeskriptor für eine Antwort

Eine Antwortnachricht enthält die folgenden Felder im Nachrichtendeskriptor:

MsgType
Dieses Feld ist MQMT_REPLY.

MsgId
Dieses Feld wird vom WS-Manager generiert.

CorrelId
Dieses Feld wird in Übereinstimmung mit den Berichtsoptionen der Befehlsnachricht generiert.

Format
Dieses Feld ist MQFMT_ADMIN.

Encoding
Setzen Sie diese Option auf MQENC_NATIVE.

CodedCharSetId
Setzen Sie diese Option auf MQCCSI_Q_MGR.

Persistence
Das gleiche wie in der Befehlsnachricht.

Priority
Das gleiche wie in der Befehlsnachricht.

Die Antwort wird mit MQPMO_PASS_IDENTITY_CONTEXT generiert.

Standardantworten

Befehlsnachrichten mit dem Headertyp MQCFT_COMMAND, Standardantworten werden generiert. Solche Befehle werden auf allen Plattformen außer z/OSunterstützt.

Es gibt drei Typen von Standardantworten:

- OK-Antwort
- Fehlerantwort
- Datenantwort

OK-Antwort

Diese Antwort besteht aus einer Nachricht, die mit einem Befehlsformat-Header beginnt und ein *CompCode* -Feld von MQCC_OK oder MQCC_WARNING enthält.

Für MQCC_OK lautet der *Reason* MQRC_NONE.

Für MQCC_WARNING identifiziert der *Reason* die Art der Warnung. In diesem Fall kann dem Befehlsformat-Header ein oder mehrere Warnparameterstrukturen folgen, die diesem Ursachencode angemessen sind.

In beiden Fällen können weitere Parameterstrukturen für einen inquire-Befehl wie in den folgenden Abschnitten beschrieben ausgeführt werden.

Fehlerantwort

Wenn der Befehl einen Fehler aufweist, werden eine oder mehrere Fehlernachrichten gesendet (mehr als eine Nachricht kann auch für einen Befehl gesendet werden, der normalerweise nur eine einzige Antwortnachricht hätte). Diese Fehlerantwortnachrichten verfügen über MQCFC_LAST oder MQCFC_NOT_LAST, die entsprechend gesetzt sind.

Jede solche Nachricht beginnt mit einem Antwortformatheader mit einem *CompCode* -Wert von MQCC_FAILED und einem Feld *Reason* , in dem der jeweilige Fehler identifiziert wird. Im Allgemeinen wird in jeder Nachricht ein anderer Fehler beschrieben. Darüber hinaus weist jede Nachricht entweder null oder eine (nie mehr als eine) Fehlerparameterstruktur nach dem Header auf. Diese Parameterstruk-

tur, sofern vorhanden, ist eine MQCFIN-Struktur, wobei ein *Parameter* -Feld eine der folgenden Werte enthält:

- MQIACF_PARAMETER_ID

  Das Feld *Value* in der Struktur ist die Parameter-ID des Fehlers, der fehlerhaft war (z. B. MQCA_Q_NAME).

- MQIACF_ERROR_ID

  Dieser Wert wird zusammen mit einem *Reason* -Wert (im Befehlsformat-Header) von MQRC_UNEXPECTED_ERROR verwendet. Das Feld *Value* in der MQCFIN-Struktur ist der unerwartete Ursachencode, der vom Befehlsserver empfangen wurde.

- MQIACF_SELECTOR

  Dieser Wert tritt auf, wenn eine mit dem Befehl gesendete Listenstruktur (MQCFIL) einen doppelten Selektor enthält oder eine nicht gültige Auswahl ist. Das Feld *Reason* im Befehlsformat-Header gibt den Fehler an, und das Feld *Value* in der MQCFIN-Struktur ist der Parameterwert in der MQCFIL-Struktur des fehlerhaft bestellten Befehls.

- MQIACF_ERROR_OFFSET

  Dieser Wert tritt auf, wenn ein Datenvergleichsfehler im Befehl Ping Channel vorliegt. Das Feld *Value* in der Struktur ist der Offset des Ping-Channel-Vergleichsfehlers.

- MQIA_CODED_CHAR_SET_ID

  Dieser Wert tritt auf, wenn die ID des codierten Zeichensatzes im Nachrichtendeskriptor der eingehenden PCF-Befehlsnachricht nicht mit der ID des Zielwarteschlangenmanagers übereinstimmt. Das Feld *Value* in der Struktur ist die ID des codierten Zeichensatzes des Warteschlangenmanagers.

Die letzte (oder einzige) Fehlerantwortnachricht ist eine Zusammenfassungsantwort mit einem *CompCode* -Feld von MQCC_FAILED und einem *Reason* -Feld von MQRCCF_COMMAND_FAILED. Diese Nachricht weist keine Parameterstruktur nach der Kopfzeile auf.

Datenantwort

Diese Antwort setzt sich aus einer OK-Antwort (wie zuvor beschrieben) in einen Befehl inquire zusammen. Auf die OK-Antwort folgen weitere Strukturen, die die angeforderten Daten enthalten, wie in Definitionen der programmierbaren Befehlsformate beschrieben.

Anwendungen dürfen nicht davon abhängig sein, dass diese zusätzlichen Parameterstrukturen in einer bestimmten Reihenfolge zurückgegeben werden.

Berechtigungsprüfung für PCF-Befehle

Wenn ein PCF-Befehl verarbeitet wird, wird die *UserIdentifier* aus dem Nachrichtendeskriptor in der Befehlsnachricht für die erforderlichen WebSphere MQ -Objektberechtigungsprüfungen verwendet. Die Berechtigungsprüfung wird auf jeder Plattform, wie in diesem Thema beschrieben, unterschiedlich implementiert.

Die Prüfungen werden auf dem System ausgeführt, auf dem der Befehl verarbeitet wird. Daher muss diese Benutzer-ID auf dem Zielsystem vorhanden sein und die erforderlichen Berechtigungen zum Verarbeiten des Befehls haben. Wenn die Nachricht von einem fernen System stammt, ist eine Möglichkeit, die auf dem Zielsystem vorhandene ID zu erreichen, eine übereinstimmende Benutzer-ID auf dem lokalen und dem fernen System zu haben.

IBM WebSphere MQ für Windows-, UNIX and Linux -Systeme

Um einen PCF-Befehl verarbeiten zu können, muss die Benutzer-ID über die *dsp* -Berechtigung für das WS-Manager-Objekt auf dem Zielsystem verfügen. Darüber hinaus werden WebSphere MQ -Objektberechtigungsprüfungen für bestimmte PCF-Befehle ausgeführt (siehe Tabelle 1 auf Seite 17).

Wenn Sie einen der folgenden Befehle verarbeiten möchten , muss die Benutzer-ID zur Gruppe *mqm* gehören.

Anmerkung: Nur unter Windows kann die Benutzer-ID zur Gruppe *Administratoren* oder zur Gruppe *mqm*gehören.

- Kanal ändern
- Kanal kopieren
- Kanal erstellen
- Kanal löschen
- Pingkanal
- Kanal zurücksetzen
- Kanal starten
- Kanal stoppen
- Kanalinitiator starten
- Kanal-Listener starten
- Kanal auflösen
- Cluster zurücksetzen
- Cluster aktualisieren
- WS-Manager aussetzen
- WS-Manager wiederaufnehmen

WebSphere MQ für HP Integrity NonStop Server

Um einen PCF-Befehl verarbeiten zu können, muss die Benutzer-ID über die *dsp* -Berechtigung für das WS-Manager-Objekt auf dem Zielsystem verfügen. Darüber hinaus werden IBM WebSphere MQ-Objektberechtigungsprüfungen für bestimmte PCF-Befehle ausgeführt, wie in Tabelle 1 auf Seite 17 dargestellt.

Gehen Sie wie folgt vor, um einen der folgenden Befehle zu verarbeiten : Die Benutzer-ID muss zur Gruppe *mqm* gehören:

- Kanal ändern
- Kanal kopieren
- Kanal erstellen
- Kanal löschen
- Pingkanal
- Kanal zurücksetzen
- Kanal starten
- Kanal stoppen
- Kanalinitiator starten
- Kanal-Listener starten
- Kanal auflösen
- Cluster zurücksetzen
- Cluster aktualisieren
- WS-Manager aussetzen
- WS-Manager wiederaufnehmen

WebSphere MQ -Objektberechtigungen

Tabelle 1. Windows, HP Integrity NonStop Server, UNIX and Linux -Systeme-Objektberechtigungen

Befehl	**WebSphere MQ -Objektberechtigung**	**Klassenberechtigung (für Objekttyp)**
Authentifizierungsinformationen ändern	dsp und chg	Nicht zutreffend
Kanal ändern	dsp und chg	Nicht zutreffend
Channel-Listener ändern	dsp und chg	Nicht zutreffend
Clientverbindungskanal ändern	dsp und chg	Nicht zutreffend
Namensliste ändern	dsp und chg	Nicht zutreffend
Prozess ändern	dsp und chg	Nicht zutreffend
Warteschlange ändern	dsp und chg	Nicht zutreffend
Warteschlangenmanager ändern	chg *siehe Anmerkung 3 und Anmerkung 5*	Nicht zutreffend
Change Service	dsp und chg	Nicht zutreffend
Warteschlange löschen	clr	Nicht zutreffend
Authentifizierungsinformationen kopieren	DSP	crt
Authentifizierungsinformationen kopieren (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* chg	crt
Kanal kopieren	DSP	crt
Copy Channel (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* chg	crt
Channel-Listener kopieren	DSP	crt
Copy Channel Listener (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* chg	crt
Clientverbindungskanal kopieren	DSP	crt
Clientverbindungskanal kopieren (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* chg	crt
Namensliste kopieren	DSP	crt
Namensliste kopieren (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* dsp und chg	crt
Prozess kopieren	DSP	crt
Prozess kopieren (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* chg	crt
Warteschlange kopieren	DSP	crt
Warteschlange kopieren (Ersetzen) *siehe Anmerkung 1*	*from:* dsp *to:* dsp und chg	crt
Authentifizierungsinformationen erstellen	*(Systemstandardauthentifizierungsinformationen)* dsp	crt

Tabelle 1. Windows, HP Integrity NonStop Server, UNIX and Linux -Systeme-Objektberechtigungen (Forts.)		
Befehl	**WebSphere MQ -Objektberechtigung**	**Klassenberechtigung (für Objekttyp)**
Authentifizierungsinformationen erstellen (Ersetzen) *siehe Anmerkung 1*	*(Systemstandardauthentifizierungsinformationen)* dsp *to:* chg	crt
Kanal erstellen	*(Systemstandardkanal)* dsp	crt
Kanal erstellen (Ersetzen) *siehe Anmerkung 1*	*(Systemstandardkanal)* dsp *to:* chg	crt
Kanallistener erstellen	*(Systemstandardlistener)* dsp	crt
Create Channel Listener (Ersetzen) *siehe Anmerkung 1*	*(system default listener)* dsp *to:* chg	crt
Clientverbindungskanal erstellen	*(Systemstandardkanal)* dsp	crt
Clientverbindungskanal erstellen (Ersetzen) *siehe Anmerkung 1*	*(Systemstandardkanal)* dsp *to:* chg	crt
Namensliste erstellen	*(Systemstandardnamensliste)* dsp	crt
Create Namelist (Ersetzen) *siehe Anmerkung 1*	*(system default namelist)* dsp *to:* dsp und chg	crt
Prozess erstellen	*(Systemstandardprozess)* dsp	crt
Erzeugen: Prozess (Ersetzen) *siehe Anmerkung 1*	*(Systemstandardprozess)* dsp *to:* chg	crt
Warteschlange erstellen	*(Systemstandardwarteschlange)* dsp	crt
Warteschlange erstellen (Ersetzen) *siehe Anmerkung 1*	*(Systemstandardwarteschlange)* dsp *to:* dsp und chg	crt
Service erstellen	*(Systemstandardwarteschlange)* dsp	crt
Service erstellen (Ersetzen) *siehe Anmerkung 1*	*(Systemstandardwarteschlange)* dsp *to:* chg	crt
Authentifizierungsinformationen löschen	dsp und dlt	Nicht zutreffend
Berechtigungssatz löschen	*(Warteschlangenmanagerobjekt)* chg *siehe Anmerkung 4*	*siehe Anmerkung 4*
Kanal löschen	dsp und dlt	Nicht zutreffend
Kanallistener löschen	dsp und dlt	Nicht zutreffend
Clientverbindungskanal löschen	dsp und dlt	Nicht zutreffend
Delete Namelist	dsp und dlt	Nicht zutreffend
Prozess löschen	dsp und dlt	Nicht zutreffend
Warteschlange löschen	dsp und dlt	Nicht zutreffend
Service löschen	dsp und dlt	Nicht zutreffend
Authentifizierungsinformationen abgefragt	DSP	Nicht zutreffend

Tabelle 1. Windows, HP Integrity NonStop Server, UNIX and Linux -Systeme-Objektberechtigungen (Forts.)		
Befehl	**WebSphere MQ -Objektberechtigung**	**Klassenberechtigung (für Objekttyp)**
Berechtigungsdatensätze anfragen	*siehe Anmerkung 4*	*siehe Anmerkung 4*
Kanalinquire	DSP	Nicht zutreffend
Channel-Listener inquire	DSP	Nicht zutreffend
Kanalstatus abfragen (für **ChannelType** MQCHT_CLSSDR)	inq	Nicht zutreffend
Clientverbindungskanal inquire	DSP	Nicht zutreffend
Namensliste inquire	DSP	Nicht zutreffend
Prozess inquire	DSP	Nicht zutreffend
Warteschlange einfragen	DSP	Nicht zutreffend
Warteschlangenmanager abfragen	*siehe Anmerkung 3*	Nicht zutreffend
Warteschlangenstatus abgefragt	DSP	Nicht zutreffend
Inquire Service	DSP	Nicht zutreffend
Pingkanal	Strg	Nicht zutreffend
Ping-WS-Manager	*siehe Anmerkung 3*	Nicht zutreffend
WS-Manager aktualisieren	(Warteschlangenmanagerobjekt) chg	Nicht zutreffend
Sicherheit aktualisieren (für **SecurityType** MQSECTYPE_SSL)	(Warteschlangenmanagerobjekt) chg	Nicht zutreffend
Kanal zurücksetzen	ctrlx	Nicht zutreffend
Warteschlangenmanager zurücksetzen	(Warteschlangenmanagerobjekt) chg	Nicht zutreffend
Warteschlangenstatistik zurücksetzen	dsp und chg	Nicht zutreffend
Kanal auflösen	ctrlx	Nicht zutreffend
Berechtigungssatz festlegen	*(Warteschlangenmanagerobjekt)* chg *siehe Anmerkung 4*	*siehe Anmerkung 4*
Kanal starten	Strg	Nicht zutreffend
Kanal stoppen	Strg	Nicht zutreffend
Stop Connection	(Warteschlangenmanagerobjekt) chg	Nicht zutreffend
Listener starten	Strg	Nicht zutreffend
Listener stoppen	Strg	Nicht zutreffend
Service starten	Strg	Nicht zutreffend
Service stoppen	Strg	Nicht zutreffend
Escapezeichen	*siehe Anmerkung 2*	*siehe Anmerkung 2*

Anmerkungen:

1. Dieser Befehl wird angewendet, wenn das zu ersetzendes Objekt vorhanden ist, andernfalls ist die Berechtigungs-Prüfung für "Erstellen" oder "Kopieren ohne Ersetzen".
2. Die erforderliche Berechtigung wird durch den MQSC-Befehl bestimmt, der durch den Escape-Text definiert wird, und entspricht einem der vorherigen Befehle.
3. Um einen PCF-Befehl verarbeiten zu können, muss die Benutzer-ID über dsp-Berechtigung für das WS-Manager-Objekt auf dem Zielsystem verfügen.
4. Dieser PCF-Befehl ist nur dann autorisiert, wenn der Befehlsserver mit dem Parameter -a gestartet wurde. Standardmäßig wird der Befehlsserver gestartet, wenn der Warteschlangenmanager gestartet wird, und ohne den Parameter '-a'. Weitere Informationen finden Sie im Handbuch zur Systemadministration.
5. Das Erteilen einer Benutzer-ID *chg* für einen Warteschlangenmanager bietet die Möglichkeit, Berechtigungsdatensätze für alle Gruppen und Benutzer festzulegen. Erteilen Sie diese Berechtigung nicht für normale Benutzer oder Anwendungen.

WebSphere MQ stellt auch einige Kanalsicherheitsexitpunkte bereit, sodass Sie eigene Benutzerexitprogramme für die Sicherheitsprüfung bereitstellen können. Einzelheiten hierzu finden Sie im Handbuch Anzeigen eines Kanals.

Verwendung von MQAI zur Vereinfachung der Verwendung von PCFs

Die MQAI ist eine Verwaltungsschnittstelle zu WebSphere MQ , die auf den Plattformen AIX, HP-UX, IBM i, Linux, Solaris und Windows verfügbar ist.

Der MQAI führt Verwaltungstasks in einem Warteschlangenmanager durch die Verwendung von *Datensäcken* aus. Mit Hilfe von Datensäcken können Sie Eigenschaften (oder Parameter) von Objekten in einer Weise bearbeiten, die einfacher ist als die Verwendung von PCFs.

Verwenden Sie die MQAI wie folgt:

Vereinfachung der Verwendung von PCF-Nachrichten
Die MQAI ist eine einfache Methode für die Verwaltung von WebSphere MQ. Sie müssen keine eigenen PCF-Nachrichten schreiben, um Probleme im Zusammenhang mit komplexen Datenstrukturen zu vermeiden.

Um Parameter in Programmen zu übergeben, die mit MQI-Aufrufen geschrieben wurden, muss die PCF-Nachricht den Befehl und Details der Zeichenfolge oder ganzzahligen Daten enthalten. Dazu benötigen Sie mehrere Anweisungen in Ihrem Programm für jede Struktur, und es muss Speicherplatz zugeordnet werden. Diese Aufgabe kann langwierig und mühselig sein.

Programme, die mit der MQAI geschrieben wurden, übergeben Parameter in den entsprechenden Datenbehälter und Sie benötigen nur eine Anweisung für jede Struktur. Die Verwendung von MQAI-Datenbehältern macht die Verarbeitung von Arrays und die Zuordnung von Speicher überflüssig und bietet eine gewisse Isolation von den Details der PCF.

Einfachere Handhabung von Fehlerbedingungen
Es ist schwierig, Rückgabecodes von PCF-Befehlen abzurufen, aber die MQAI erleichtert dem Programm die Behandlung von Fehlerbedingungen.

Nachdem Sie Ihren Datenbehälter erstellt und gefüllt haben, können Sie mit dem Aufruf `mqExecute` , der auf Antwortnachrichten wartet, eine Verwaltungsbefehlsnachricht an den Befehlsserver eines Warteschlangenmanagers senden. Der Aufruf `mqExecute` verarbeitet den Austausch mit dem Befehlsserver und gibt Antworten in einem *Antwortbehälter*zurück.

Weitere Informationen zur MQAI finden Sie unter „Einführung in die IBM WebSphere MQ Administration Interface (MQAI)” auf Seite 21.

Einführung in die IBM WebSphere MQ Administration Interface (MQAI)

Die IBM WebSphere MQ -Verwaltungsschnittstelle (MQAI) ist eine Programmierschnittstelle für IBM WebSphere MQ. Sie führt Verwaltungstasks in einem IBM WebSphere MQ-Warteschlangenmanager mithilfe von Datenbehältern aus, um Eigenschaften (oder Parameter) von Objekten auf eine Art und Weise zu handhaben, die einfacher ist als die Verwendung von PCFs (Programmable Command Formats).

MQAI-Konzepte und -Terminologie

Die MQAI ist eine Programmierschnittstelle zu WebSphere MQunter Verwendung der Programmiersprache C und auch Visual Basic für Windows. Sie ist auf anderen Plattformen als z/OSverfügbar.

Er führt Verwaltungstasks für einen WebSphere MQ mithilfe von Datenbehältern aus. Mit Datenbehältern können Eigenschaften (oder Parameter) von Objekten einfacher bearbeitet werden als unter Verwendung von Programmable Command Formats (PCFs). PCFs lassen sich mit der MQAI leichter bearbeiten als mit den Aufrufen MQGET und MQPUT.

Weitere Informationen zu Datenbehältern finden Sie unter „Datenbehälter” auf Seite 47. Weitere Informationen zu PCFs finden Sie unter „Einführung in Programmierbare Befehlsformate” auf Seite 9

Verwendung der MQAI

Sie können die MQAI für folgende Tasks verwenden:

- Vereinfachte Verwendung von PCF-Nachrichten. Die MQAI stellt eine einfache Methode zur Verwaltung von WebSphere MQdar. Sie müssen keine eigenen PCF-Nachrichten schreiben und so die Probleme im Zusammenhang mit komplexen Datenstrukturen vermeiden.
- Vereinfachte Fehlerbehandlung. Es ist schwierig, Rückgabecodes von den MQSC-Scriptbefehlen (WebSphere MQ) abzurufen, aber die MQAI erleichtert dem Programm die Behandlung von Fehlerbedingungen.
- Datenaustausch zwischen Anwendungen. Die Anwendungsdaten werden im PCF-Format gesendet und von der MQAI gepackt und entpackt. Wenn Ihre Nachrichtendaten aus Ganzzahlen und Zeichenfolgen bestehen, können Sie die MQAI verwenden, um die Vorteile der integrierten WebSphere MQ -Datenkonvertierung für PCF-Daten zu nutzen. Auf diese Weise wird vermieden, dass Datenkonvertierungsexits geschrieben werden müssen. Weitere Informationen zur Verwendung von MQAI zur Verwaltung von WebSphere MQ und zum Austausch von Daten zwischen Anwendungen finden Sie unter „Verwendung von MQAI zur Vereinfachung der Verwendung von PCFs” auf Seite 20.

Beispiele für die Verwendung der MQAI

Die angezeigte Liste enthält einige Beispielprogramme, die die Verwendung von MQAI veranschaulichen. Die Beispiele führen die folgenden Tasks aus:

1. Erstellen Sie eine lokale Warteschlange. „C-Beispielprogramm zum Erstellen einer lokalen Warteschlange (amqsaicq.c)” auf Seite 22
2. Ereignisse in der Anzeige mit einem einfachen Ereignismonitor anzeigen.„C-Beispielprogramm für die Anzeige von Ereignissen mit einem Ereignismonitor (amqsaiem.c)” auf Seite 26
3. Drucken Sie eine Liste mit allen lokalen Warteschlangen und deren aktuellen Tiefen aus.„C-Beispielprogramm für die Abfrage von Warteschlangen und Druckinformationen (amqsailq.c)” auf Seite 38
4. Drucken Sie eine Liste aller Kanäle und deren Typen aus.„C-Beispielprogramm für die Abfrage von Kanalobjekten (amqsaicl.c)” auf Seite 33

Erstellen Ihrer MQAI-Anwendung

Um Ihre Anwendung mithilfe der MQAI zu erstellen, stellen Sie eine Verknüpfung zu denselben Bibliotheken her wie für WebSphere MQ. Informationen zum Erstellen Ihrer WebSphere MQ -Anwendungen finden Sie unter WebSphere MQ -Anwendung erstellen .

Hinweise und Tipps für die Konfiguration von WebSphere MQ mit MQAI

Die MQAI verwendet PCF-Nachrichten, um Verwaltungsbefehle an den Befehlsserver zu senden, anstatt direkt mit dem Befehlsserver selbst zu arbeiten. Tipps für die Konfiguration von WebSphere MQ mithilfe der MQAI finden Sie unter „Hinweise und Tipps zur Konfiguration von IBM WebSphere MQ” auf Seite 42

IBM WebSphere MQ -Verwaltungsschnittstelle (MQAI)

IBM WebSphere MQ für Windows, AIX, Linux, HP-UXund Solaris unterstützen die IBM WebSphere MQ -Verwaltungsschnittstelle (MQAI). Bei der MQAI handelt es sich um eine Programmierschnittstelle zu IBM WebSphere MQ, die eine Alternative zur MQI für das Senden und Empfangen von PCFs darstellt.

Die MQAI verwendet *Datenbehälter*, mit denen Sie Eigenschaften (oder Parameter) von Objekten einfacher bearbeiten können, als dies mit der direkten Verwendung von PCFs über MQAI möglich wäre.

Die MQAI stellt einen vereinfachten Programmierzugriff auf PCF-Nachrichten bereit, indem Parameter in den Datenbehälter übergeben werden, damit für jede Struktur nur eine Anweisung erforderlich ist. Durch diesen Zugriff muss der Programmierer keine Arrays bearbeiten und keinen Speicherbereich zuordnen und es wird eine gewisse Isolation von den Details von PCF erreicht.

Die MQAI verwaltet WebSphere MQ , indem sie PCF-Nachrichten an den Befehlsserver sendet und auf eine Antwort wartet.

Die MQAI wird im zweiten Teil dieses Handbuchs beschrieben. Eine Beschreibung einer Objektmodellschnittstelle der Komponente für die MQAI finden Sie in der Java verwenden-Dokumentation.

C-Beispielprogramm zum Erstellen einer lokalen Warteschlange (amqsaicq.c)

Das Beispiel-C-Programm `amqsaicq.c` erstellt mithilfe der MQAI eine lokale Warteschlange.

```
/**********************************************************************/
/*                                                                    */
/* 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);

}
```

C-Beispielprogramm für die Anzeige von Ereignissen mit einem Ereignismonitor (amqsaiem.c)

Das Beispiel-C-Programm amqsaiem.c veranschaulicht einen Basisereignismonitor mithilfe der 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, 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;
}
```

C-Beispielprogramm für die Abfrage von Kanalobjekten (amqsaicl.c)

Das Beispiel-C-Programm `amqsaicl.c` untersucht Kanalobjekte mit der 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);
}
```

C-Beispielprogramm für die Abfrage von Warteschlangen und Druckinformationen (amqsailq.c)

Das Beispiel-C-Programm amqsailq.c stellt die aktuelle Tiefe der lokalen Warteschlangen mit der MQAI (MQAI) in Frage.

```
/******************************************************************************/
/*                                                                            */
/* 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);
}
```

Hinweise und Tipps zur Konfiguration von IBM WebSphere MQ

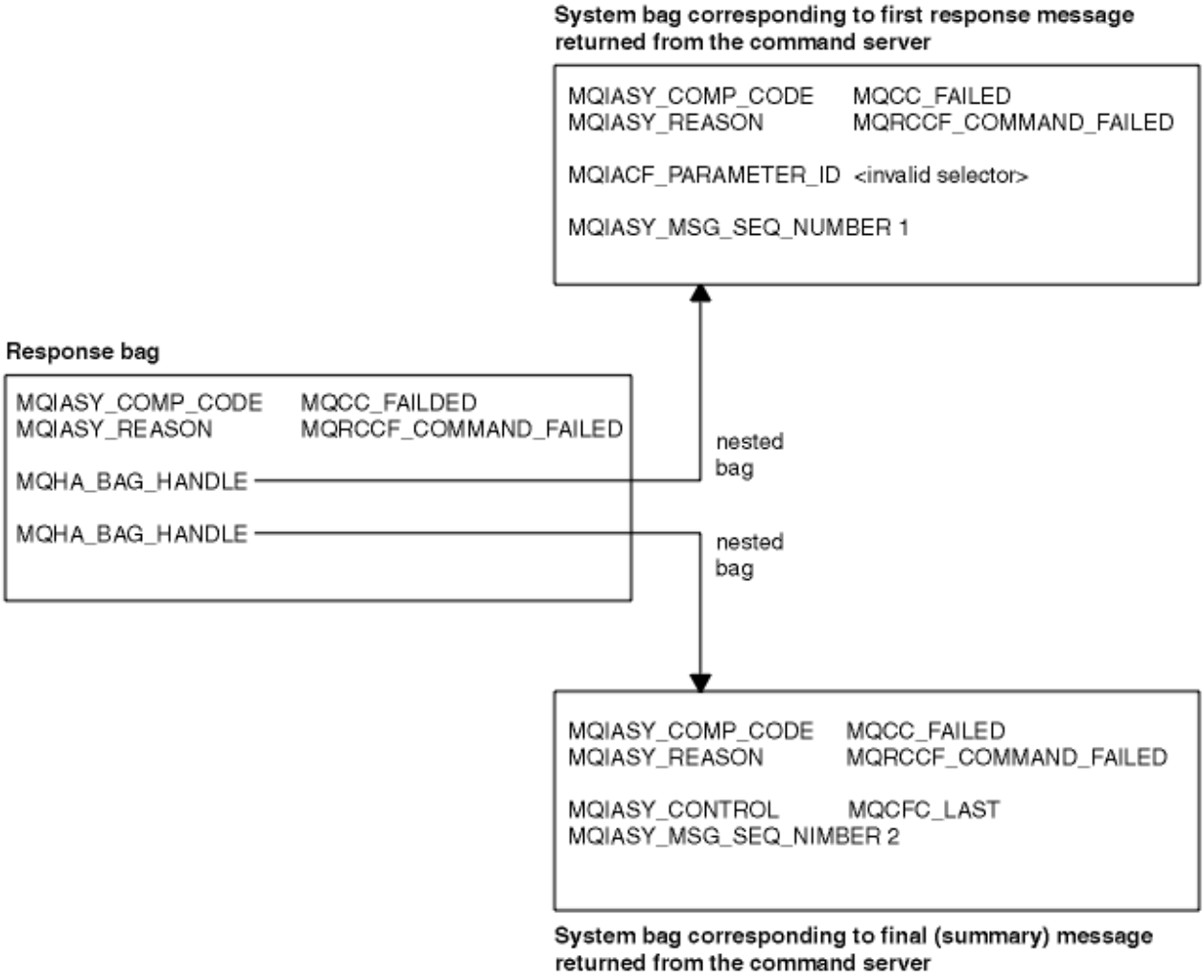
Hinweise und Tipps zur Programmierung bei Verwendung der MQAI.

Die MQAI verwendet PCF-Nachrichten, um Verwaltungsbefehle an den Befehlsserver zu senden, anstatt direkt mit dem Befehlsserver selbst zu arbeiten. Im Folgenden finden Sie einige Tipps für die Konfiguration von WebSphere MQ mithilfe der MQAI:

- Zeichenfolgen in WebSphere MQ werden mit Leerzeichen auf eine feste Länge aufgefüllt. Bei Verwendung von C können auf null endende Zeichenfolgen normalerweise als Eingabeparameter für WebSphere MQ -Programmierschnittstellen bereitgestellt werden.
- Wenn Sie den Wert eines Zeichenfolgeattributs löschen möchten, setzen Sie ihn auf ein einzelnes Leerzeichen und nicht auf eine leere Zeichenfolge.
- Berücksichtigen Sie im Voraus die Attribute, die Sie ändern möchten, und fragen Sie nur diese Attribute ab.
- Bestimmte Attribute können nicht geändert werden, z. B. ein Warteschlangenname oder ein Kanaltyp. Stellen Sie sicher, dass Sie versuchen, nur die Attribute zu ändern, die geändert werden können. Weitere Informationen finden Sie in der Liste der erforderlichen und optionalen Parameter für das jeweilige PCF-Änderungsobjekt. Siehe Definitionen der programmierbaren Befehlsformate .
- Wenn ein MQAI-Aufruf fehlschlägt, wird ein Teil des Fehlers an den Antwortbehälter zurückgegeben. Weitere Details können dann in einem verschachtelten Sack gefunden werden, auf den der Selektor MQHA_BAG_HANDLE zugreifen kann. Wenn z. B. ein MQExecute-Aufruf mit dem Ursachencode MQRCCF_COMMAND_FAILED fehlschlägt, wird diese Information in der Antworttasche zurückgegeben. Eine mögliche Ursache für diesen Ursachencode ist, dass ein angegebener Selektor für den Typ der Befehlsnachricht nicht gültig ist und dass diese Detailinformationen in einem verschachtelten Sack gefunden werden, auf den ein Taschengriff zugreifen kann.

  Weitere Informationen zu MQExecute finden Sie in „Verwaltungsbefehle mit dem Aufruf 'mqExecute' an den Befehlsserver senden” auf Seite 57.

  Das folgende Diagramm zeigt dieses Szenario:



Erweiterte MQAI-Themen

Informationen zur Indexierung, Datenkonvertierung und Verwendung des Nachrichtendeskriptors

- Indexierung

  Indizes werden verwendet, wenn vorhandene Datenelemente aus einem Behälter ersetzt oder entfernt werden, um die Einfügefolge zu erhalten. Vollständige Details zur Indexierung finden Sie unter „Indexierung in der MQAI” auf Seite 43.

- Datenkonvertierung

  Die in einem MQAI-Datenbehälter enthaltenen Zeichenfolgen können in einer Vielzahl von codierten Zeichensätzen enthalten sein und diese können mit dem Aufruf 'mqSetInteger' konvertiert werden. Vollständige Details zur Datenkonvertierung finden Sie unter „Datenkonvertierung in der MQAI” auf Seite 45.

- Verwendung des Nachrichtendeskriptors

  MQAI generiert einen Nachrichtendeskriptor, der bei der Erstellung des Datensacks auf einen Anfangswert gesetzt wird. Vollständige Details zur Verwendung des Nachrichtendeskriptors finden Sie unter „Verwendung des Nachrichtendeskriptors in der MQAI” auf Seite 46.

Indexierung in der MQAI

Indizes werden zum Ersetzen oder Entfernen bereits vorhandener Datenelemente in einem Behälter verwendet. Es gibt drei Möglichkeiten zur Indexierung, mit denen Datenelemente auf einfache Weise abgerufen werden können.

Jedem Selektor und Wert eines Datenelements in einem Behälter werden drei Indexnummern zugeordnet:

- Der Index relativ zu anderen Elementen mit demselben Selektor.
- Der Index relativ zur Selektorkategorie (Benutzer oder System), zu der das Element gehört.
- Der Index relativ zu allen Datenelementen im Behälter (Benutzer und System).

Dies ermöglicht ein Indexieren nach Benutzerselektoren, Systemselektoren oder beiden Selektortypen, wie in Abbildung 1 auf Seite 44 dargestellt.

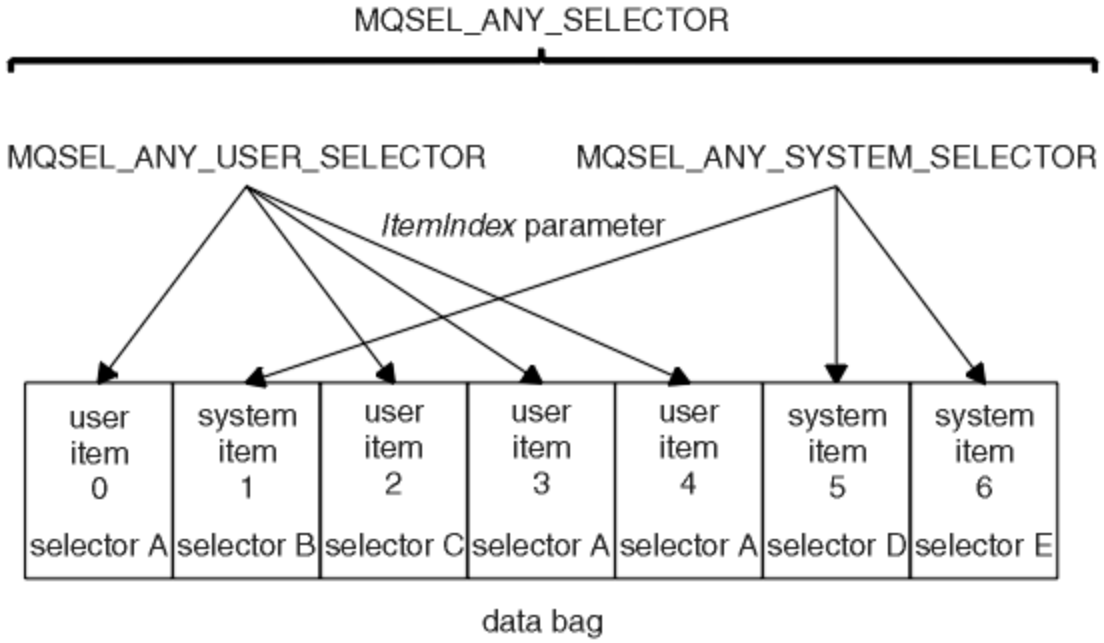


Abbildung 1. Indexierung

In Abbildung 1 auf Seite 44 kann mit den folgenden Indexpaaren auf Benutzerelement 3 (Selektor A) verwiesen werden:

Selector	***ItemIndex***
Selektor A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	3

Der Index basiert auf null, genau wie eine Gruppe in C. Bei "n" Vorkommen reicht der Index lückenlos von null bis "n-1".

Indizes werden zum Ersetzen oder Entfernen bereits vorhandener Datenelemente in einem Behälter verwendet. Diese Verwendungsweise wirkt sich nicht auf die Einfügereihenfolge, aber möglicherweise auf die Indizes anderer Datenelemente aus. Beispiele hierzu finden Sie in den Abschnitten Informationen in einem Behälter ändern und Datenelemente ändern.

Durch die drei Indexierungstypen lassen sich Datenelemente leicht abrufen. Wenn in einem Behälter beispielsweise drei Instanzen eines bestimmten Selektors vorkommen, kann mit dem Aufruf "mqCountItems" die Anzahl der Instanzen dieses Selektors gezählt werden und mit den Aufrufen vom Typ "mqInquire*" können sowohl der Selektor als auch der Index angegeben werden, um nur diese Werte abzufragen. Diese Vorgehensweise eignet sich für Attribute, die eine Werteliste aufweisen können, z. B. einige Exits von Kanälen.

Datenkonvertierung in der MQAI

Die in einem MQAI-Datenbehälter enthaltenen Zeichenfolgen können in einer Vielzahl von codierten Zeichensätzen enthalten sein. Diese Zeichenfolgen können mit dem Aufruf 'mqSetInteger' konvertiert werden.

Genau wie PCF-Nachrichten können auch die Zeichenfolgen in einem MQAI-Datenbehälter verschiedene codierte Zeichensätze aufweisen. Normalerweise gilt für alle Zeichenfolgen in einer PCF-Nachricht derselbe codierte Zeichensatz, der dem Zeichensatz des Warteschlangenmanagers entspricht.

Jedes Zeichenfolgeelement in einem Datenbehälter enthält zwei Werte: die Zeichenfolge selbst und die ID des codierten Zeichensatzes (CCSID). Die Zeichenfolge, die zum Behälter hinzugefügt wird, stammt aus dem Parameter *Buffer* des Aufrufs "mqAddString" oder "mqSetString". Die CCSID stammt aus dem Systemelement, das den Selektor MQIASY_CODED_CHAR_SET_ID enthält. Diese ID wird als *Behälter-CCSID* bezeichnet und kann mit dem Aufruf "mqSetInteger" geändert werden.

Wenn Sie den Wert einer Zeichenfolge abrufen, die sich in einem Datenbehälter befindet, stellt die CCSID einen Ausgabeparameter des Aufrufs dar.

In Tabelle 2 auf Seite 45 sind die Regeln dargestellt, die beim Konvertieren von Datenbehältern in Nachrichten und umgekehrt angewendet werden:

Tabelle 2. CCSID-Verarbeitung

MQAI-Aufruf	CCSID	Eingabe des Aufrufs	Ausgabe des Aufrufs
mqBagToBuffer	Behälter-CCSID (1)	Wird ignoriert.	Nicht geändert
mqBagToBuffer	Zeichenfolge-CCSIDs in Behälter	Genutzt	Nicht geändert
mqBagToBuffer	Zeichenfolge-CCSIDs in Puffer	Nicht zutreffend	Kopiert aus Zeichenfolge-CCSIDs in Behälter
mqBufferToBag	Behälter-CCSID (1)	Wird ignoriert.	Nicht geändert
mqBufferToBag	Zeichenfolge-CCSIDs in Puffer	Genutzt	Nicht geändert
mqBufferToBag	Zeichenfolge-CCSIDs in Behälter	Nicht zutreffend	Kopiert aus Zeichenfolge-CCSIDs in Puffer
mqPutBag	MQMD-CCSID	Genutzt	Nicht geändert (2)
mqPutBag	Behälter-CCSID (1)	Wird ignoriert.	Nicht geändert
mqPutBag	Zeichenfolge-CCSIDs in Behälter	Genutzt	Nicht geändert
mqPutBag	Zeichenfolge-CCSIDs in gesendeter Nachricht	Nicht zutreffend	Kopiert aus Zeichenfolge-CCSIDs in Behälter
mqGetBag	MQMD-CCSID	Für Datenkonvertierung der Nachricht verwendet	Auf CCSID der zurückgegebenen Daten gesetzt (3)
mqGetBag	Behälter-CCSID (1)	Wird ignoriert.	Nicht geändert
mqGetBag	Zeichenfolge-CCSIDs in Nachricht	Genutzt	Nicht geändert
mqGetBag	Zeichenfolge-CCSIDs in Behälter	Nicht zutreffend	Kopiert aus Zeichenfolge-CCSIDs in Nachricht
mqExecute	Anforderungsbehälter-CCSID	Für MQMD der Anforderungsnachricht verwendet (4)	Nicht geändert

Tabelle 2. CCSID-Verarbeitung (Forts.)			
MQAI-Aufruf	**CCSID**	**Eingabe des Aufrufs**	**Ausgabe des Aufrufs**
mqExecute	Antwortbehälter-CCSID	Für Datenkonvertierung der Antwortnachricht verwendet (4)	Auf CCSID der zurückgegebenen Daten gesetzt (3)
mqExecute	Zeichenfolge-CCSIDs in Anforderungsbehälter	Für Anforderungsnachricht verwendet	Nicht geändert
mqExecute	Zeichenfolge-CCSIDs in Antwortbehälter	Nicht zutreffend	Kopiert aus Zeichenfolge-CCSIDs in Antwortnachricht

Anmerkungen:

1. Die Behälter-CCSID ist das Systemelement mit dem Selektor MQIASY_CODED_CHAR_SET_ID.
2. MQCCSI_Q_MGR wird in die tatsächliche CCSID des Warteschlangenmanagers geändert.
3. Wenn eine Datenkonvertierung angefordert wird, entspricht die CCSID der zurückgegebenen Daten dem Ausgabewert. Wenn keine Datenkonvertierung angefordert wird, entspricht die CCSID der zurückgegebenen Daten dem Nachrichtenwert. Wenn eine Datenkonvertierung angefordert wird, aber fehlschlägt, wird keine Nachricht zurückgegeben.
4. Wenn die CCSID den Wert MQCCSI_DEFAULT aufweist, wird die CCSID des Warteschlangenmanagers verwendet.

Verwendung des Nachrichtendeskriptors in der MQAI

Der von der MQAI generierte Nachrichtendeskriptor ist bei der Erstellung des Datenbehälters auf einen Anfangswert gesetzt.

Der PCF-Befehlstyp stammt aus dem Systemelement mit dem Selektor MQIASY_TYPE. Beim Erstellen eines Datenbehälters wird der Anfangswert dieses Elements je nach Typ des erstellten Behälters festgelegt:

Tabelle 3. PCF-Befehlstyp	
Behältertyp	**Anfangswert des Elements MQIASY_TYPE**
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_*	MQCFT_USER

Wenn die MQAI einen Nachrichtendeskriptor generiert, hängen die für die Parameter *Format* und *MsgType* verwendeten Werte vom Wert des Systemelements mit dem Selektor MQIASY_TYPE ab, wie in Tabelle 3 auf Seite 46 dargestellt.

Tabelle 4. Parameter "Format" und "MsgType" des Nachrichtendeskriptors		
PCF-Befehlstyp	**Format**	**MsgType**
MQCFT_COMMAND	MQFMT_ADMIN	MQMT_REQUEST
MQCFT_REPORT	MQFMT_ADMIN	MQMT_REPORT
MQCFT_RESPONSE	MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	MQFMT_ADMIN	MQMT_DATAGRAM
MQCFT_EVENT	MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_*	MQFMT_PCF	MQMT_DATAGRAM

Tabelle 4 auf Seite 46 zeigt Folgendes: Wenn Sie einen Verwaltungs- oder Befehlsbehälter erstellen, hat der Parameter *Format* des Nachrichtendeskriptors den Wert MQFMT_ADMIN und *MsgType* den Wert MQMT_REQUEST. Dies eignet sich, wenn eine PCF-Anforderungsnachricht an den Befehlsserver gesendet und eine Antwort erwartet wird.

Die anderen Parameter im Nachrichtendeskriptor nehmen die in Tabelle 5 auf Seite 47 dargestellten Werte an.

Tabelle 5. Werte des Nachrichtendeskriptors

Parameter	**Wert**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	Informationen hierzu finden Sie unter Tabelle 4 auf Seite 46.
Expiry	30 Sekunden (Anmerkung „1” auf Seite 47)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	Je nach Behälter-CCSID (Anmerkung „2” auf Seite 47)
Format	Informationen hierzu finden Sie unter Tabelle 4 auf Seite 46.
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	Siehe Anmerkung „3” auf Seite 47
ReplyToQMgr	Leer

Anmerkungen:

1. Dieser Wert kann im Aufruf **mqExecute** mit dem Parameter **OptionsBag** überschrieben werden. Informationen hierzu finden Sie unter **mqExecute**.
2. Siehe „Datenkonvertierung in der MQAI” auf Seite 45.
3. Name der benutzerdefinierten Antwortwarteschlange oder MQAI-generierten temporären dynamischen Warteschlange für Nachrichten des Typs **MQMT_REQUEST**. Andernfalls leer.

Datenbehälter

Ein Datenbehälter wird verwendet, um Eigenschaften oder Parameter von Objekten mit der MQAI zu bearbeiten.

Datenbehälter

- Der Datenbehälter enthält null oder mehr *Datenelemente* . Diese Datenelemente werden innerhalb des Beutels bestellt, wenn sie in die Tasche gesteckt werden. Dies wird als *Einfügeanordnung* bezeichnet.

Jedes Datenelement enthält einen *Selektor* , der das Datenelement identifiziert, und einen *Wert* dieses Datenelements, das entweder eine ganze Zahl, eine 64-Bit-Ganzzahl, ein ganzzahliger Filter, eine Zeichenfolge, ein Zeichenfolgefilter, eine Bytefolge, ein Bytefolgefilter oder ein Handle eines anderen Beutels sein kann. Datenelemente werden in „Datenelemente” auf Seite 50 detailliert beschrieben.

Es gibt zwei Typen von Selektoren: *Benutzerselektoren* und *Systemselektoren* . Diese werden in MQAI Selectors beschrieben. Die Selektoren sind in der Regel eindeutig, aber es ist möglich, mehrere Werte für denselben Selektor zu haben. In diesem Fall gibt ein *Index* das jeweilige Vorkommen des Selektors an, das erforderlich ist. Indizes werden unter „Indexierung in der MQAI” auf Seite 43 beschrieben.

Eine Hierarchie dieser Konzepte ist in Abbildung 1dargestellt.

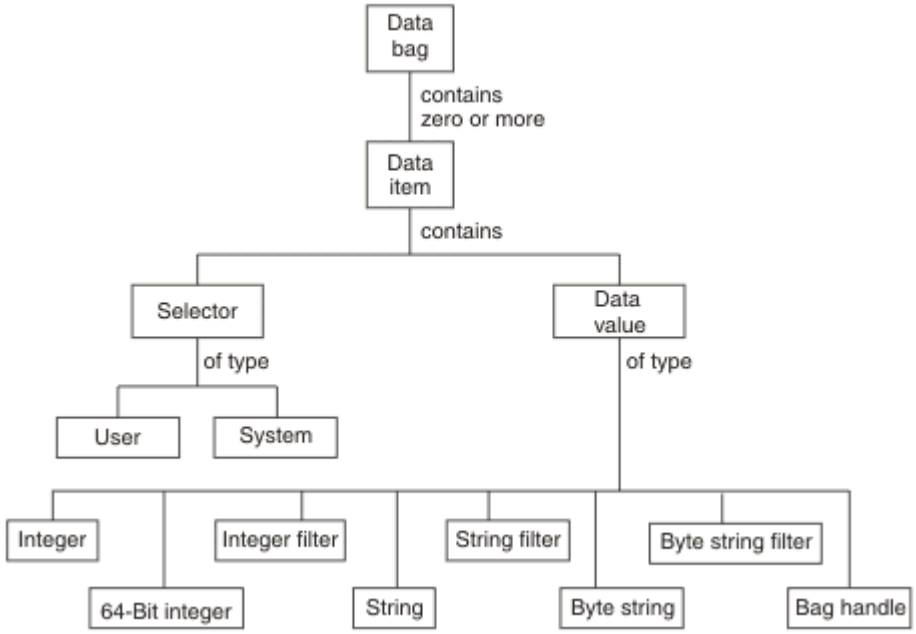


Abbildung 2. Hierarchiehierarchie von MQAI-Konzepten

Die Hierarchie wurde in einem vorherigen Abschnitt erläutert.

Typen von Datenbehälter

Sie können den Datenbehältertyp, den Sie erstellen möchten, je nach der Task, die sie ausführen wollen, wählen:

Benutzerbehälter
Eine einfache Tasche, die für Benutzerdaten verwendet wird.

Verwaltungstasche
Ein Behälter, der für Daten erstellt wurde, die zum Verwalten von WebSphere MQ -Objekten verwendet werden, indem Verwaltungsnachrichten an einen Befehlsserver gesendet werden. Der Verwaltungsbehälter schließt automatisch bestimmte Optionen ein, wie unter „Datenstaschen erstellen und löschen” auf Seite 49 beschrieben wird.

Befehlsbehälter
Ein Behälter, der auch für Befehle zur Verwaltung von WebSphere MQ -Objekten erstellt wurde. Im Gegensatz zum Verwaltungsbehälter impliziert der Befehlsbehälter jedoch nicht automatisch bestimmte Optionen, obwohl diese Optionen verfügbar sind. Weitere Informationen zu Optionen finden Sie in „Datenstaschen erstellen und löschen” auf Seite 49.

Gruppenbehälter
Ein Behälter, der zum Speichern einer Gruppe gruppierter Datenelemente verwendet wird. Gruppenbehälter können nicht für die Verwaltung von WebSphere MQ -Objekten verwendet werden.

Darüber hinaus wird die **Systemtasche** von der MQAI erstellt, wenn eine Antwortnachricht vom Befehlsserver zurückgegeben und in die Ausgabetasche eines Benutzers gestellt wird. Ein Systembehälter kann vom Benutzer nicht geändert werden.

Datenbeutel verwenden Die verschiedenen Möglichkeiten zur Verwendung von Datensäcken sind in diesem Thema aufgeführt:

Daten-Bags verwenden

Die verschiedenen Arten der Verwendung von Datensäcken werden in der folgenden Liste angezeigt:

- Sie können Datenbehälter erstellen und löschen „Datenstaschen erstellen und löschen” auf Seite 49.
- Sie können Daten zwischen Anwendungen mit Hilfe von Datenbeuteln senden „Datenbehälter einreihen und empfangen” auf Seite 50.
- Sie können Datenbehältern Datenelemente hinzufügen „Datenelemente zu Behältern hinzufügen” auf Seite 51.
- Sie können einen Abfragebefehl in einem Datenbehälter „Abfragebefehl zu einem Behälter hinzufügen” auf Seite 52 hinzufügen.
- Sie können innerhalb von Datenbehältern Abfragen vornehmen „Inquiring innerhalb von Datensäcken” auf Seite 53.
- Sie können Datenelemente innerhalb von Datenbehältern zählen „Datenelemente zählen” auf Seite 55.
- Sie können Informationen in einem Datenbehälter ändern „Informationen in einer Tasche ändern” auf Seite 53.
- Sie können einen Datenbehälter leeren „Inhalt einer Tasche mit dem mqClearBag-Aufruf löschen” auf Seite 54.
- Sie können einen Datenbehälter abschneiden „Abschneiden eines Behälters mit dem mqTruncateBag-Aufruf” auf Seite 54.
- Sie können Behälter und Puffer konvertieren „Konvertieren von Taschen und Puffern” auf Seite 55.

Datenstaschen erstellen und löschen

Datenstaschen erstellen

Wenn Sie die MQAI verwenden möchten, erstellen Sie zuerst einen Datenbehälter mit dem mqCreateBag-Aufruf. Als Eingabe für diesen Aufruf geben Sie eine oder mehrere Optionen an, um die Erstellung der Tasche zu steuern.

Mit dem Parameter *Options* des MQCreateBag-Aufrufs können Sie auswählen, ob eine Benutzergruppe, eine Befehlstasche, ein Gruppenbehälter oder ein Verwaltungsbehälter erstellt werden soll.

Zum Erstellen eines Benutzerbeutels, einer Befehlstasche oder eines Gruppenbeutels können Sie eine oder mehrere weitere Optionen auswählen, um:

- Verwenden Sie das Listenformular, wenn zwei oder mehr benachbarte Vorkommen desselben Selektors in einer Tasche vorhanden sind.
- Sie können die Datenelemente neu anordnen, wenn sie einer PCF-Nachricht hinzugefügt werden, um sicherzustellen, dass die Parameter in der richtigen Reihenfolge sind. Weitere Informationen zu Datenelementen finden Sie im Abschnitt „Datenelemente” auf Seite 50.
- Überprüfen Sie die Werte von Benutzerselektoren für Elemente, die Sie der Tasche hinzufügen.

Verwaltungssäcke implizieren automatisch diese Optionen.

Ein Datenbehälter wird durch den Griff identifiziert. Die Kennung des Sackes wird von mqCreateBag zurückgegeben und muss bei allen anderen Aufrufen, die den Datenbehälter verwenden, bereitgestellt werden.

Eine vollständige Beschreibung des mqCreateBag-Aufrufs finden Sie unter mqCreateBag .

Datentaschen löschen

Alle Datenbehälter, die vom Benutzer erstellt werden, müssen auch mit dem mqDeleteBag-Aufruf gelöscht werden. Wenn z. B. eine Tasche im Benutzercode erstellt wird, muss sie auch im Benutzercode gelöscht werden.

Systemtaschen werden automatisch von der MQAI erstellt und gelöscht. Weitere Informationen hierzu finden Sie unter „Verwaltungsbefehle mit dem Aufruf 'mqExecute' an den Befehlsserver senden” auf Seite 57. Der Benutzercode kann eine Systemhülle nicht löschen.

Eine vollständige Beschreibung des Aufrufs 'mqDeleteBag' finden Sie unter mqDeleteBag .

Datenbehälter einreihen und empfangen

Daten können auch zwischen Anwendungen gesendet werden, indem Datensäcke mithilfe der mqPutBag- und mqGetBag-Aufrufe gestellt und erhalten werden. Auf diese Weise kann die MQAI den Puffer und nicht die Anwendung verarbeiten. Der mqPutBag-Aufruf konvertiert den Inhalt des angegebenen Sacks in eine PCF-Nachricht und sendet die Nachricht an die angegebene Warteschlange, und der mqGetBag-Aufruf entfernt die Nachricht aus der angegebenen Warteschlange und konvertiert sie zurück in einen Datenbehälter. Daher ist der mqPutBag-Aufruf das Äquivalent des Aufrufs mqBagToBuffer, gefolgt von MQPUT, und der mqGetBag entspricht dem MQGET-Aufruf gefolgt von mqBufferToBag.

Weitere Informationen zum Senden und Empfangen von PCF-Nachrichten in einer bestimmten Warteschlange finden Sie in „PCF-Nachrichten in einer angegebenen Warteschlange senden und empfangen” auf Seite 12.

Anmerkung: Wenn Sie sich für die Verwendung des mqGetBag-Aufrufs entscheiden, müssen die PCF-Details in der Nachricht korrekt sein. Wenn dies nicht der Fall ist, werden die entsprechenden Fehlerergebnisse angezeigt und die PCF-Nachricht wird nicht zurückgegeben.

Datenelemente

Datenelemente werden zum Füllen von Datenbehältern bei deren Erstellung verwendet. Bei diesen Datenelementen kann es sich um Benutzer- oder Systemelemente handeln.

Diese Benutzerelemente enthalten Benutzerdaten, wie z. B. Attribute von Objekten, die verwaltet werden. Systemelemente sollten für mehr Kontrolle über die generierten Nachrichten verwendet werden: z. B. die Generierung von Nachrichtenheadern. Weitere Informationen zu Systemelementen finden Sie in „Systemelemente” auf Seite 51.

Typen von Datenelementen

Wenn Sie einen Datenbehälter erstellt haben, können Sie ihn mit Ganzzahl-oder Zeichenfolgeelementen füllen. Sie können alle drei Typen von Elementen abgefragt werden.

Das Datenelement kann eine ganze Zahl oder ein Zeichenfolgeelement sein. Im Folgenden sind die Typen der Datenelemente in der MQAI verfügbar:

- Integer
- 64-Bit-Ganzzahl
- Filter für Ganzzahl
- Zeichenfolge
- Zeichenfolgefilter
- Bytefolge
- Filter für Bytefolge
- Behälterkennung

Datenelemente verwenden

Dies sind die folgenden Arten der Verwendung von Datenelementen:

- „Datenelemente zählen” auf Seite 55.
- „Datenelemente löschen” auf Seite 56.
- „Datenelemente zu Behältern hinzufügen” auf Seite 51.
- „Datenelemente filtern und abfragen” auf Seite 52.

Systemelemente

Systemelemente können für Folgendes verwendet werden:

- Die Generierung von PCF-Headern. Systemelemente können die PCF-Befehls-ID, die Steueroptionen, die Nachrichtenfolgenummer und den Befehlstyp steuern.
- Datenkonvertierung. Systemelemente behandeln die Zeichensatzes-ID für die Zeichenfolge-Elemente in der Tasche.

Wie alle Datenelemente bestehen Systemelemente aus einem Selektor und einem Wert. Weitere Informationen zu diesen Selektoren und zu ihrer Verwendung finden Sie unter MQAI-Selektoren .

Systemelemente sind eindeutig. Ein oder mehrere Systemelemente können durch einen Systemselektor identifiziert werden. Es gibt nur ein Vorkommen jedes Systemselektors.

Die meisten Systemelemente können geändert werden (siehe „Informationen in einer Tasche ändern” auf Seite 53), aber die Optionen zur Erstellung von Behältern können vom Benutzer nicht geändert werden. Systemelemente können nicht gelöscht werden. (Siehe „Datenelemente löschen” auf Seite 56.)

Datenelemente zu Behältern hinzufügen

Beim Erstellen eines Datenbehälters können Sie diesen mit Datenelementen füllen. Bei diesen Datenelementen kann es sich um Benutzer- oder Systemelemente handeln. Weitere Informationen zu Datenelementen finden Sie im Abschnitt „Datenelemente” auf Seite 50.

Mit der Option MQAI können Sie ganze Zahlen, 64-Bit-Ganzzahlelemente, Ganzzahlfilterelemente, Zeichenfolgeelemente, Zeichenfolgefilter, Bytefolgeelemente und Bytefolgefilterelemente zu Taschen hinzufügen. Dies ist in Abbildung 3 auf Seite 51 dargestellt. Die Elemente werden durch einen Selektor identifiziert. Gewöhnlich identifiziert ein Selektor nur ein Element, aber dies ist nicht immer der Fall. Wenn ein Datenelement mit dem angegebenen Selektor bereits in der Tasche vorhanden ist, wird dem Ende der Tasche eine zusätzliche Instanz dieses Selektors hinzugefügt.

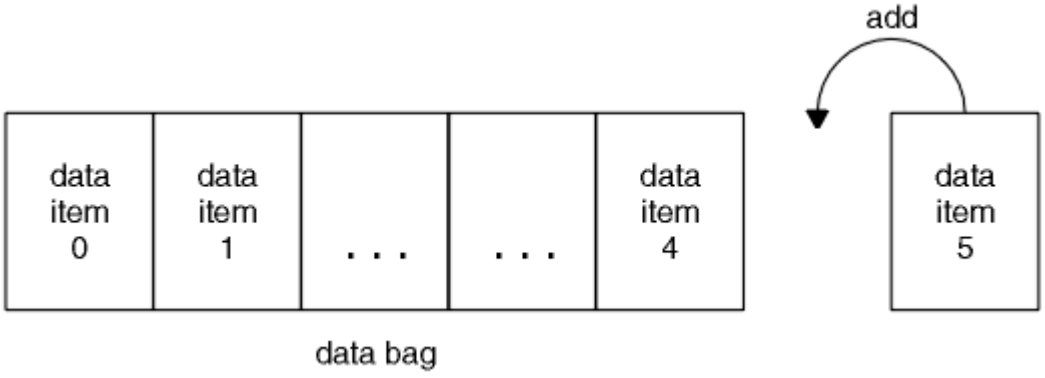


Abbildung 3. Datenelemente hinzufügen

Fügen Sie mit den mqAdd * -Aufrufen Datenelemente zu einer Tasche hinzu:

- Um ganzzahlige Elemente hinzuzufügen, verwenden Sie den mqAddInteger-Aufruf wie in mqAddInteger beschrieben.
- Um 64-Bit-Integer-Elemente hinzuzufügen, verwenden Sie den mqAddInteger64-Aufruf wie in mqAddInteger64 beschrieben.

- Um ganzzahlige Filterelemente hinzuzufügen, verwenden Sie den mqAddIntegerFilter-Aufruf wie in mqAddIntegerFilter beschrieben.
- Um Zeichenfolgeelemente hinzuzufügen, verwenden Sie den mqAddString-Aufruf wie in mqAddString beschrieben.
- Um Zeichenfolgefilterelemente hinzuzufügen, verwenden Sie den mqAddStringFilter-Aufruf wie in mqAddStringFilter beschrieben.
- Um Bytefolgeelemente hinzuzufügen, verwenden Sie den mqAddByteString-Aufruf wie in mqAddByteString beschrieben.
- Um Bytefolgefilterelemente hinzuzufügen, verwenden Sie den mqAddByteStringFilter-Aufruf wie in mqAddByteStringFilter beschrieben.

Weitere Informationen zum Hinzufügen von Datenelementen zu einem Behälter finden Sie im Abschnitt „Systemelemente” auf Seite 51.

Abfragebefehl zu einem Behälter hinzufügen

Der Aufruf 'mqAddInquiry' wird verwendet, um einen Abfragebefehl zu einem Behälter hinzuzufügen. Der Aufruf ist speziell für Verwaltungszwecke, so dass er nur mit Verwaltungsbehältern verwendet werden kann. Hier können Sie die Selektoren von Attributen angeben, die Sie von WebSphere MQabfragen möchten.

Eine vollständige Beschreibung des Aufrufs 'mqAddInquiry' finden Sie unter mqAddInquiry .

Datenelemente filtern und abfragen

Wenn Sie die MQAI verwenden, um die Attribute von WebSphere MQ -Objekten abzufragen, können Sie die Daten, die an Ihr Programm zurückgegeben werden, auf zwei Arten steuern.

- Sie können die Daten, die mit den Aufrufen 'mqAddInteger' und 'mqAddString' zurückgegeben werden, ***filter*** . Mit dieser Methode können Sie ein Paar aus *Selector* und *ItemValue* angeben, z. B.:

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

  In diesem Beispiel wird angegeben, dass der Warteschlangentyp (*Selector*) lokal (*ItemValue*) sein muss und dass diese Spezifikation mit den Attributen des Objekts (in diesem Fall eine Warteschlange) übereinstimmen muss, über das/die Sie abgefragt werden.

  Andere Attribute, die gefiltert werden können, entsprechen den PCF Inquire*-Befehlen, die unter „Einführung in Programmierbare Befehlsformate” auf Seite 9 beschrieben sind. Wenn Sie beispielsweise die Attribute eines Kanals abgefragt haben, lesen Sie den Befehl Inquire Channel in dieser Produktdokumentation. Die Parameter "Required parameters" und "Optional parameters" des Befehls "Inquire Channel" geben die Selektoren an, die Sie zum Filtern verwenden können.
- Mithilfe des Aufrufs 'mqAddInquiry' können Sie bestimmte Attribute eines Objekts ***abfragen*** . Gibt den Selektor an, an dem Sie interessiert sind. Wenn Sie den Selektor nicht angeben, werden alle Attribute des Objekts zurückgegeben.

Im Folgenden finden Sie ein Beispiel für das Filtern und Abfragen der Attribute einer Warteschlange:

```
/* 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, ...)
```

Weitere Beispiele zum Filtern und Abfragen von Datenelementen finden Sie im Abschnitt „Beispiele für die Verwendung der MQAI” auf Seite 21.

Inquiring innerhalb von Datensäcken

Sie können Folgendes einfragen:

- Der Wert eines ganzzahligen Elements mit Hilfe des Aufrufs 'mqInquireInteger'. Siehe mqInquireInteger .
- Der Wert eines ganzzahligen 64-Bit-Elements mit dem Aufruf 'mqInquireInteger64'. Siehe mqInquireInteger64 .
- Der Wert eines Ganzzahlfilterelements mit dem Aufruf 'mqInquireIntegerFilter'. Siehe mqInquireIntegerFilter .
- Der Wert eines Zeichenfolgeelements, das mit dem Aufruf 'mqInquireString' verwendet wird. Siehe mqInquireString .
- Der Wert eines Zeichenfolgefilterelements mit dem Aufruf 'mqInquireStringFilter'. Siehe mqInquireStringFilter .
- Der Wert eines Bytefolgeelements mit dem Aufruf 'mqInquireByteString'. Siehe mqInquireByteString .
- Der Wert eines Bytefolgefilterelements mit Hilfe des Aufrufs 'mqInquireByteStringFilter'. Siehe mqInquireByteStringFilter .
- Der Wert einer Taschenkennung, die mit dem mqInquireBag-Aufruf verwendet wird. Siehe mqInquireBag .

Sie können auch den Typ (Ganzzahl, 64-Bit-Integer, Ganzzahlfilter, Zeichenfolge, Zeichenfolgefilter, Bytefolge, Bytefolgefilter oder Taschengriff) eines bestimmten Elements mit dem Aufruf mqInquireItemInfo abgefragt werden. Siehe mqInquireItemInfo.

Informationen in einer Tasche ändern

Mit dem MQAI können Sie Informationen innerhalb einer Tasche mit Hilfe der mqSet * -Aufrufe ändern. Sie haben folgende Möglichkeiten:

1. Datenelemente in einer Tasche ändern. Der Index ermöglicht das Ersetzen einer einzelnen Instanz eines Parameters, indem das Vorkommen des zu änderbaren Elements angegeben wird (siehe Abbildung 4 auf Seite 53).

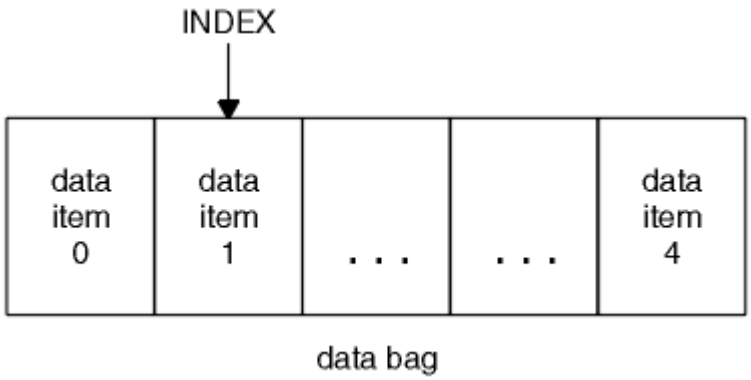


Abbildung 4. Ein einzelnes Datenelement ändern

2. Löschen Sie alle vorhandenen Vorkommen des angegebenen Selektors und fügen Sie dem Ende des Sacks ein neues Vorkommen hinzu. (Siehe Abbildung 5 auf Seite 54.) Ein spezieller Indexwert ermöglicht ***alle*** Instanzen eines Parameters, der ersetzt werden soll.

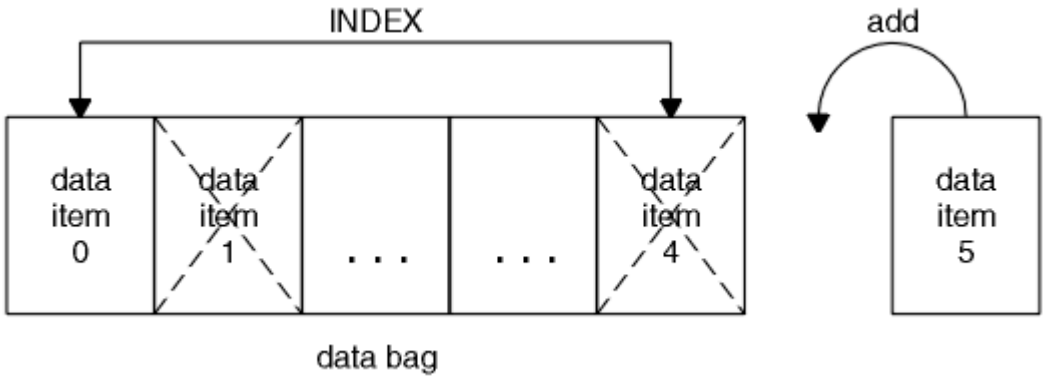


Abbildung 5. Alle Datenelemente ändern

Anmerkung: Der Index erfüllt die Einfügefolge innerhalb des Sacks, kann aber die Indizes anderer Datenelemente beeinflussen.

Mit dem mqSetInteger-Aufruf können Sie Integer-Elemente in einer Tasche ändern. Mit dem Aufruf mqSetInteger64 können Sie 64-Bit-Integer-Elemente ändern. Mit dem mqSetIntegerFilter-Aufruf können Sie ganzzahlige Filterelemente ändern. Mit dem mqSetString-Aufruf können Sie Zeichenfolgeelemente ändern. Mit dem mqSetStringFilter-Aufruf können Sie Zeichenfolgefilterelemente ändern. Mit dem mqSetByteString-Aufruf können Sie Bytefolgenelemente ändern. Mit dem Aufruf 'mqSetByteStringFilter' können Sie Bytefolgefilterelemente ändern. Alternativ können Sie diese Aufrufe verwenden, um alle vorhandenen Vorkommen des angegebenen Selektors zu löschen und am Ende der Tasche ein neues Vorkommen hinzuzufügen. Bei dem Datenelement kann es sich um ein Benutzerelement oder um ein Systemelement handeln.

Eine vollständige Beschreibung dieser Aufrufe finden Sie unter:

- mqSetInteger
- mqSetInteger64
- mqSetIntegerFilter
- mqSetString
- mqSetStringFilter
- mqSetByteString
- mqSetByteStringFilter

Inhalt einer Tasche mit dem mqClearBag-Aufruf löschen

Mit dem mqClearBag-Aufruf werden alle Benutzerelemente aus einem Benutzerbehälter entfernt und die Systemelemente werden auf ihre Anfangswerte zurückversetzt. Die in der Tasche enthaltenen Systemtaschen werden ebenfalls gelöscht.

Eine vollständige Beschreibung des mqClearBag-Aufrufs finden Sie unter mqClearBag .

Abschneiden eines Behälters mit dem mqTruncateBag-Aufruf

Der mqTruncateBag-Aufruf reduziert die Anzahl der Benutzerelemente in einem Benutzerbehälter, indem er die Elemente vom Ende der Tasche aus löscht, beginnend mit dem zuletzt hinzugefügten Element. Es kann beispielsweise verwendet werden, wenn dieselben Headerinformationen verwendet werden, um mehr als eine Nachricht zu generieren.

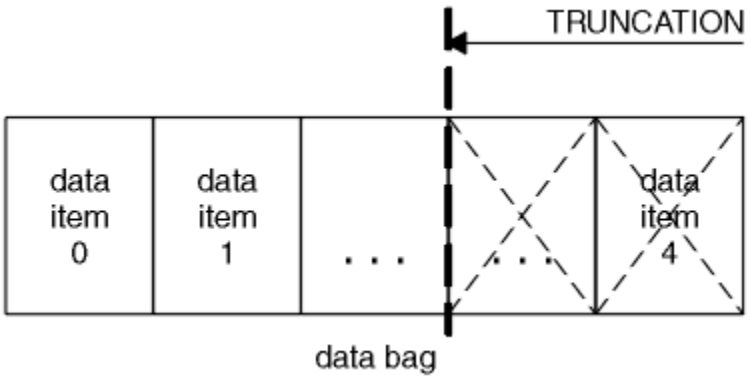


Abbildung 6. Einen Behälter abschneiden

Eine vollständige Beschreibung des mqTruncateBag-Aufrufs finden Sie unter mqTruncateBag .

Konvertieren von Taschen und Puffern

Um Daten zwischen Anwendungen zu senden, werden zuerst die Nachrichtendaten in eine Tasche gestellt. Anschließend werden die Daten in der Tasche mit Hilfe des mqBagToBuffer-Aufrufs in eine PCF-Nachricht konvertiert. Die PCF-Nachricht wird unter Verwendung des MQPUT-Aufrufs an die erforderliche Warteschlange gesendet. Dies ist in Abbildung Abbildung 7 auf Seite 55 dargestellt. Eine vollständige Beschreibung des Aufrufs 'mqBagToBuffer' finden Sie in mqBagToBuffer .

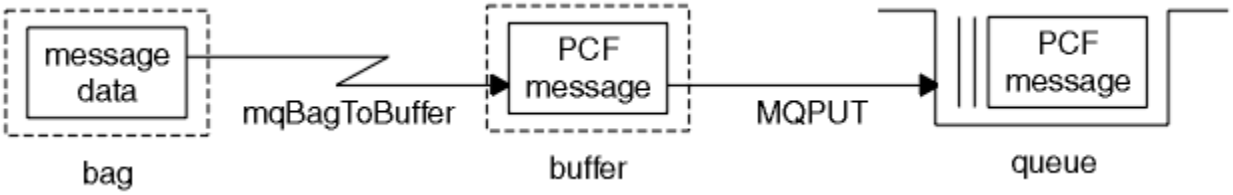


Abbildung 7. Konvertieren von Taschen in PCF-Nachrichten

Um Daten zu empfangen, wird die Nachricht mit dem MQGET-Aufruf in einen Puffer empfangen. Die Daten im Puffer werden dann mit Hilfe des mqBufferToBag-Aufrufs in eine Tasche konvertiert, so dass der Puffer eine gültige PCF-Nachricht enthält. Dies wird in Abbildung Abbildung 8 auf Seite 55 gezeigt. Eine vollständige Beschreibung des mqBufferToBag-Aufrufs finden Sie unter mqBufferToBag .

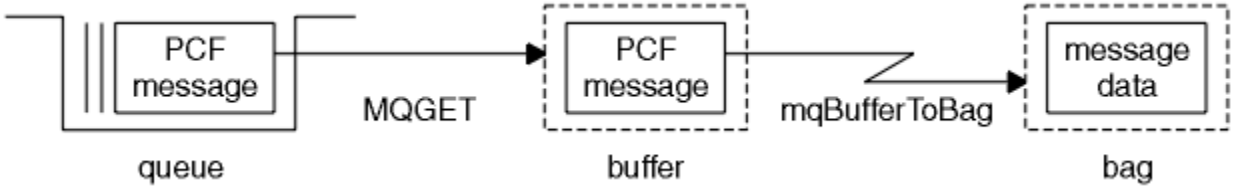


Abbildung 8. PCF-Nachrichten in Beutelformular umwandeln

Datenelemente zählen

Der Aufruf 'mqCountItems' zählt die Anzahl der Benutzerelemente, Systemelemente oder beides, die in einem Datenbehälter gespeichert sind, und gibt diese Zahl zurück. Beispiel: `mqCountItems(`*Bag*`, 7, ...)` gibt die Anzahl der Elemente im Behälter mit einem Selektor von 7 zurück. Sie kann Elemente durch einzelne Selektoren, durch Benutzerselektoren, durch Systemselektoren oder durch alle Selektoren zählen.

Anmerkung: Dieser Aufruf zählt die Anzahl der Datenelemente, nicht die Anzahl der eindeutigen Selektoren in der Tasche. Ein Selektor kann mehrfach vorkommen, so dass es in der Tasche weniger eindeutige Selektoren geben kann als Datenelemente.

Eine vollständige Beschreibung des Aufrufs 'mqCountItems' finden Sie unter mqCountItems .

Datenelemente löschen

Sie können Artikel aus Taschen auf verschiedene Arten löschen. Sie haben folgende Möglichkeiten:

- Entfernt ein oder mehrere Benutzerelemente aus einer Tasche. Ausführliche Informationen hierzu finden Sie unter „Datenelemente aus einer Tasche mit dem mqDeleteItem-Aufruf löschen” auf Seite 56.
- Löschen Sie ***alle*** -Benutzerelemente aus einer Tasche, d. a. *clear* , eine Tasche. Weitere Informationen finden Sie unter „Inhalt einer Tasche mit dem mqClearBag-Aufruf löschen” auf Seite 54.
- Löschen Sie die Benutzereinträge am Ende eines Sacks, d. a. *Abschneiden* einer Tasche. Ausführliche Informationen hierzu finden Sie unter „Abschneiden eines Behälters mit dem mqTruncateBag-Aufruf” auf Seite 54.

Datenelemente aus einer Tasche mit dem mqDeleteItem-Aufruf löschen

Mit dem mqDeleteItem-Aufruf werden ein oder mehrere Benutzerelemente aus einer Tasche entfernt. Der Index wird zum Löschen verwendet:

1. Ein einzelnes Vorkommen des angegebenen Selektors. (Siehe Abbildung 9 auf Seite 56.)

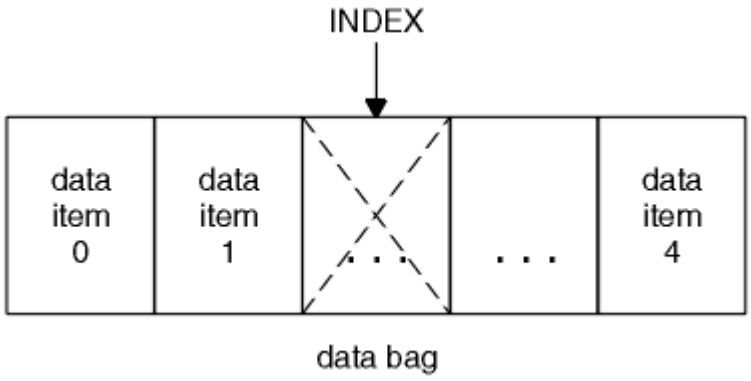


Abbildung 9. Löschen eines einzelnen Datenelements

oder

2. Alle Vorkommen des angegebenen Selektors. (Siehe Abbildung 10 auf Seite 56.)

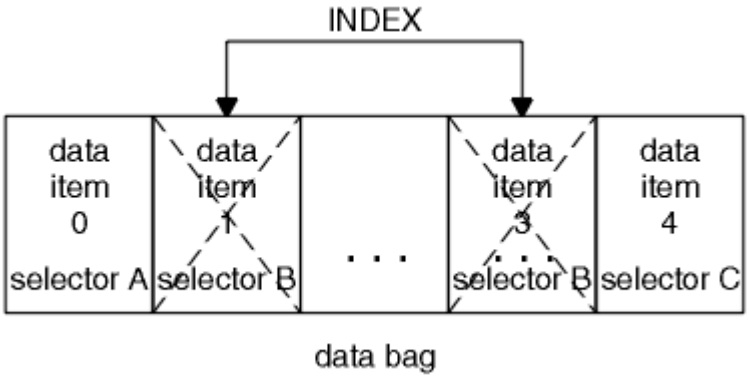


Abbildung 10. Alle Datenelemente löschen

Anmerkung: Der Index erfüllt die Einfügefolge innerhalb des Sacks, kann aber die Indizes anderer Datenelemente beeinflussen. Der Aufruf 'mqDeleteItem' beispielsweise behält die Indexwerte der Datenelemente, die dem gelöschten Element folgen, nicht bei, da die Indizes neu organisiert werden, um die Lücke zu füllen, die vom gelöschten Element übrig bleibt.

Eine vollständige Beschreibung des Aufrufs 'mqDeleteItem' finden Sie unter mqDeleteItem .

Verwaltungsbefehle mit dem Aufruf 'mqExecute' an den Befehlsserver senden

Wenn ein Datenbehälter erstellt und gefüllt wurde, kann eine Verwaltungsbefehlsnachricht mit Hilfe des Aufrufs mqExecute an den Befehlsserver eines Warteschlangenmanagers gesendet werden. Dadurch wird der Austausch mit dem Befehlsserver durchgeführt und die Antworten in einem Behälter zurückgegeben.

Nachdem Sie Ihren Datenbehälter erstellt und gefüllt haben, können Sie eine Verwaltungsbefehlsnachricht an den Befehlsserver eines Warteschlangenmanagers senden. Dies ist der einfachste Weg, indem Sie den Aufruf mqExecute verwenden. Der mqExecute-Aufruf sendet eine Verwaltungsbefehlsnachricht als nicht persistente Nachricht und wartet auf alle Antworten. Antworten werden in einem Antwortbehälter zurückgegeben. Diese können Informationen zu Attributen enthalten, die sich beispielsweise auf mehrere WebSphere MQ -Objekte oder auf eine Reihe von PCF-Fehlerantwortnachrichten beziehen. Daher kann der Antwortbehälter nur einen Rückkehrcode enthalten oder er könnte *verschachtelte Taschen* enthalten.

Antwortnachrichten werden in Systemtaschen gestellt, die vom System erstellt werden. Für Rückfragen zu den Namen von Objekten wird beispielsweise ein Systembehälter erstellt, in dem diese Objektnamen enthalten sind, und die Tasche wird in die Benutzertasche eingefügt. Handles zu diesen Taschen werden dann in die Antworttasche eingefügt und die verschachtelte Tasche kann von der Selektor MQHA_BAG_HANDLE aufgerufen werden. Der Systembehälter verbleibt im Speicher, wenn er nicht gelöscht wird, bis der Antwortbehälter gelöscht wird.

Das Konzept der *Verschachterung* ist in Abbildung 11 auf Seite 57 dargestellt.

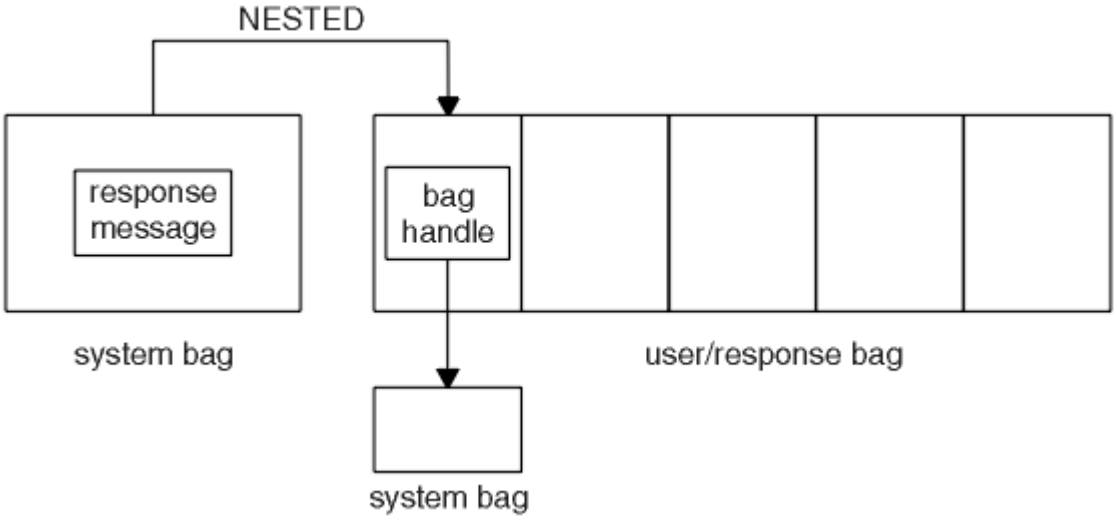


Abbildung 11. Verschachteln

Als Eingabe für den mqExecute-Aufruf müssen Sie Folgendes angeben:

- Eine MQI-Verbindungskennung.
- Der Befehl, der ausgeführt werden soll. Hierbei sollte es sich um einen der MQCMD_ * -Werte handeln.

  Anmerkung: Wenn dieser Wert von der MQAI nicht erkannt wird, wird der Wert trotzdem akzeptiert. Wenn jedoch der mqAddInquiry-Aufruf zum Einfügen von Werten in die Tasche verwendet wurde, muss dieser Parameter ein INQUIRE-Befehl sein, der von der MQAI erkannt wird. Das heißt, der Parameter muss das Format MQCMD_INQUIRE_ * haben.
- Optional ein Handle der Tasche, die Optionen enthält, die die Verarbeitung des Aufrufs steuern. Hier können Sie auch die maximale Zeit in Millisekunden angeben, die der MQAI auf jede Antwortnachricht warten soll.
- Ein Handle des Verwaltungsbeutels, der Details zum Verwaltungsbefehl enthält, der ausgegeben werden soll.
- Ein Handle des Antwortbehälters, der die Antwortnachrichten empfängt.

Folgende Optionen sind optional:

- Eine Objektkennung der Warteschlange, in die der Verwaltungsbefehl gestellt werden soll.

  Wenn keine Objektkennung angegeben ist, wird der Verwaltungsbefehl in die Warteschlange SYSTEM.ADMIN.COMMAND.QUEUE des derzeit verbundenen Warteschlangenmanagers gestellt. Dies ist die Standardeinstellung.
- Eine Objektkennung der Warteschlange, in die Antwortnachrichten gestellt werden sollen.

  Sie können auswählen, dass die Antwortnachrichten in eine dynamische Warteschlange, die automatisch von der MQAI erstellt wird, platziert werden sollen. Die erstellte Warteschlange ist nur für die Dauer des Aufrufs vorhanden und wird durch die MQAI beim Verlassen des mqExecute-Aufrufs gelöscht.

Beispiele für die Verwendung des mqExecute-Aufrufs finden Sie im Abschnitt Beispielcode.

Verwaltung mit IBM WebSphere MQ Explorer

IBM WebSphere MQ Explorer ermöglicht Ihnen die lokale oder ferne Verwaltung Ihres Netzes nur von einem Computer aus, auf dem Windowsoder Linux ausgeführt wird (x86 -und x86-64 -Plattformen).

IBM WebSphere MQ für Windowsund IBM WebSphere MQ für Linux (Plattformenx86 und x86-64) stellen eine Verwaltungsschnittstelle namens IBM WebSphere MQ Explorer bereit, um Verwaltungstasks als Alternative zur Verwendung von Steuer-oder MQSC-Befehlen auszuführen. Befehlssätze vergleichen zeigt, was Sie mit dem IBM WebSphere MQ Explorer tun können.

Mit IBM WebSphere MQ Explorer können Sie die lokale oder ferne Verwaltung Ihres Netzes von einem Computer aus ausführen, auf dem Windowsoder Linux (x86-64 -Plattformen) ausgeführt wird, indem Sie IBM WebSphere MQ Explorer auf die Warteschlangenmanager und Cluster verweisen, für die Sie sich interessieren. Die Plattformen und Versionen von IBM WebSphere MQ, die mit IBM WebSphere MQ Explorer verwaltet werden können, werden im Abschnitt „Ferne Warteschlangenmanager” auf Seite 59 beschrieben.

Informationen, wie ferne IBM WebSphere MQ-Warteschlangenmanager konfiguriert werden müssen, damit sie mit IBM WebSphere MQ Explorer verwaltet werden können, finden Sie im Abschnitt „Softwarevoraussetzungen und Definitionen” auf Seite 60.

Es ermöglicht Ihnen, Tasks auszuführen, die in der Regel mit dem Einrichten und Optimieren der Arbeitsumgebung für IBM WebSphere MQverbunden sind, entweder lokal oder über Fernzugriff innerhalb einer Systemdomäne Windows oder Linux (x86 -und x86-64 -Plattformen).

Unter Linux kann der IBM WebSphere MQ Explorer möglicherweise nicht gestartet werden, wenn Sie mehr als eine Eclipse-Installation verwenden. Wenn dies der Fall ist, starten Sie den IBM WebSphere MQ Explorer mit einer anderen Benutzer-ID als der für die andere Eclipse-Installation verwendeten.

Unter Linuxmüssen Sie zum erfolgreichen Starten von IBM WebSphere MQ Explorer in der Lage sein, eine Datei in Ihr Ausgangsverzeichnis zu schreiben, und das Ausgangsverzeichnis muss vorhanden sein.

Was Sie mit IBM WebSphere MQ Explorer tun können

Dies ist eine Liste der Tasks, die Sie mit dem IBM WebSphere MQ Explorerausführen können.

Mit IBM WebSphere MQ Explorer haben Sie folgende Möglichkeiten:

- Erstellen und löschen Sie einen WS-Manager (nur auf Ihrer lokalen Maschine).
- Starten und stoppen Sie einen WS-Manager (nur auf Ihrer lokalen Maschine).
- Definieren, Anzeigen und Ändern der Definitionen von WebSphere MQ -Objekten wie Warteschlangen und Kanälen.
- Durchsuchen Sie die Nachrichten in einer Warteschlange.
- Starten und stoppen Sie einen Kanal.
- Statusinformationen zu einem Kanal, einem Listener, einer Warteschlange oder Serviceobjekten anzeigen.

- Anzeigen von Warteschlangenmanagern in einem Cluster.
- Überprüfen Sie, welche Anwendungen, Benutzer oder Kanäle eine bestimmte Warteschlange geöffnet haben.
- Erstellen Sie mit dem Assistenten *Neuen Cluster erstellen* einen neuen WS-Manager-Cluster.
- Fügen Sie einen Warteschlangenmanager mit dem Assistenten *WS-Manager zum Cluster hinzufügen* zu einem Cluster hinzu.
- Verwalten des Authentifizierungsdatenobjekts für Secure Sockets Layer-Kanalsicherheit
- Kanalinitiatoren, Auslösemonitore und Empfangsprogramme erstellen und löschen.
- Starten oder stoppen Sie die Befehlsserver, Kanalinitiatoren, Auslösemonitore und Empfangsprogramme.
- Legen Sie bestimmte Services fest, die beim Start eines Warteschlangenmanagers automatisch gestartet werden sollen.
- Ändern der Merkmale von Warteschlangenmanagern.
- Ändern Sie den lokalen Standardwarteschlangenmanager.
- Rufen Sie die GUI von ikeyman auf, um SSL-Zertifikate (Secure Sockets Layer) zu verwalten, Zertifikate mit Warteschlangenmanagern zu verknüpfen und Zertifikatsspeicher zu konfigurieren und zu konfigurieren (nur auf Ihrer lokalen Maschine).
- JMS-Objekte aus WebSphere MQ -Objekten und WebSphere MQ -Objekte aus JMS-Objekten erstellen.
- Erstellen Sie eine JMS-Verbindungsfactory für alle derzeit unterstützten Typen.
- Ändern Sie die Parameter für jeden Service, wie z. B. die TCP-Portnummer für einen Listener oder einen Namen für die Kanalinitiatorwarteschlange.
- Den Service-Trace starten oder stoppen.

Sie führen Verwaltungstasks mit einer Reihe von *Inhaltsansichten* und *Eigenschaftendialogen* aus.

Inhaltsansicht
Eine Inhaltsansicht ist eine Anzeige, in der Folgendes angezeigt werden kann:

- Attribute und Verwaltungsoptionen in Bezug auf WebSphere MQ selbst.
- Attribute und Verwaltungsoptionen, die sich auf ein oder mehrere zugehörige Objekte beziehen.
- Attribute und Verwaltungsoptionen für einen Cluster.

Eigenschaftendialoge
Ein Eigenschaftendialog ist eine Anzeige, in der Attribute angezeigt werden, die sich auf ein Objekt in einer Reihe von Feldern beziehen, von denen einige bearbeitet werden können.

Sie navigieren in WebSphere MQ Explorer über die *NavigatoransichtNavigator*. Im Navigator können Sie die von Ihnen benötigten Inhaltsansicht auswählen.

Ferne Warteschlangenmanager

Es gibt zwei Ausnahmen für die unterstützten Warteschlangenmanager, zu denen eine Verbindung hergestellt werden kann.

Von einem Windows -oder Linux -System (x86 -und x86-64 -Plattformen) kann der WebSphere MQ Explorer mit den folgenden Ausnahmen eine Verbindung zu allen unterstützten Warteschlangenmanagern herstellen:

- WebSphere MQ für z/OS -Warteschlangenmanager vor Version 6.0.
- Derzeit unterstützte Warteschlangenmanager MQSeries V2 .

Der IBM WebSphere MQ Explorer handhabt die Unterschiede zwischen den verschiedenen Befehlsebenen und Plattformen. Wenn jedoch ein Attribut erkannt wird, das es nicht erkennt, wird das Attribut nicht angezeigt.

Wenn Sie einen Warteschlangenmanager mit V6.0 oder höher unter Windows über Fernzugriff verwalten möchten, indem Sie IBM WebSphere MQ Explorer auf einem Computer mit WebSphere MQ V5.3 verwen-

den, müssen Sie Fixpack 9 (CSD9) oder höher auf Ihrem Computer mit WebSphere MQ für Windows V5.3 installieren.

Wenn Sie einen V5.3 -Warteschlangenmanager unter iSeries über Fernzugriff verwalten möchten, verwenden Sie WebSphere MQ Explorer auf einem Computer mit WebSphere MQ V6.0 oder höher, Sie müssen Fixpack 11 (CSD11) oder höher auf Ihrem WebSphere MQ for iSeries V5.3 -Computer installieren. Dieses Fixpack behebt Verbindungsprobleme zwischen WebSphere MQ Explorer und dem Warteschlangenmanager iSeries .

Entscheiden, ob IBM WebSphere MQ Explorer verwendet werden soll

Beachten Sie bei der Entscheidung, ob der IBM WebSphere MQ Explorer bei Ihrer Installation verwendet werden soll, die in diesem Abschnitt aufgeführten Informationen.

Sie müssen die folgenden Punkte beachten:

Objektnamen
Wenn Sie in IBM WebSphere MQ Explorer Namen in Kleinbuchstaben für Warteschlangenmanager und andere Objekte verwenden, müssen Sie bei der Arbeit mit den Objekten mithilfe von MQSC-Befehlen die Objektnamen in einfache Anführungszeichen einschließen, da sie andernfalls nicht von WebSphere MQ erkannt werden.

Große Warteschlangenmanager
Der IBM WebSphere MQ Explorer funktioniert am besten mit kleinen Warteschlangenmanagern. Wenn Sie eine große Anzahl von Objekten auf einem einzelnen Warteschlangenmanager haben, können Verzögerungen auftreten, während der WebSphere MQ Explorer die erforderlichen Informationen extrahiert, um sie in einer Ansicht darzustellen.

Cluster
WebSphere MQ -Cluster können potenziell Hunderte oder Tausende Warteschlangenmanager enthalten. Der WebSphere MQ Explorer stellt die Warteschlangenmanager in einem Cluster in einer Baumstruktur dar. Die physische Größe eines Clusters hat keine Auswirkungen auf die Geschwindigkeit des IBM WebSphere MQ Explorer, da der IBM WebSphere MQ Explorer erst dann eine Verbindung zu den Warteschlangenmanagern im Cluster herstellt, wenn Sie sie auswählen.

IBM WebSphere MQ Explorer einrichten

In diesem Abschnitt werden die erforderlichen Schritte zum Einrichten des IBM WebSphere MQ Explorers beschrieben.

- „Softwarevoraussetzungen und Definitionen” auf Seite 60
- „Sicherheit” auf Seite 61
- „Warteschlangenmanager und Cluster anzeigen und ausblenden” auf Seite 65
- „Clusterzugehörigkeit” auf Seite 66
- „Datenkonvertierung” auf Seite 66

Softwarevoraussetzungen und Definitionen

Stellen Sie sicher, dass die folgenden Anforderungen erfüllt sind, bevor Sie versuchen, den IBM WebSphere MQ Explorer zu verwenden.

Der IBM WebSphere MQ Explorer kann nur über das TCP/IP-Kommunikationsprotokoll eine Verbindung zu fernen Warteschlangenmanagern herstellen.

Überprüfen Sie Folgendes:

1. Ein Befehlsserver wird auf jedem über Remotezugriff verwalteten Warteschlangenmanager ausgeführt.
2. Ein geeignetes TCP/IP-Listener-Objekt muss auf jedem fernen WS-Manager ausgeführt werden. Dieses Objekt kann der IBM WebSphere MQ-Listener oder, auf Systemen unter UNIX and Linux, der Dämon Inetd sein.

3. Ein Serververbindungskanal, der standardmäßig mit dem Namen SYSTEM.ADMIN.SVRCONN bezeichnet wird, ist auf allen fernen Warteschlangenmanagern vorhanden.

   Sie können den Kanal mit dem folgenden MQSC-Befehl erstellen:

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

   Mit diesem Befehl wird eine Basiskanaldefinition erstellt. Wenn Sie eine komplexere Definition wünschen (z. B. die Sicherheit konfigurieren), benötigen Sie zusätzliche Parameter. Weitere Informationen finden Sie unter DEFINE CHANNEL .
4. Die Systemwarteschlange (SYSTEM.MQEXPLORER.REPLY.MODEL) muss vorhanden sein.

Sicherheit

Wenn Sie WebSphere MQ in einer Umgebung verwenden, in der es wichtig ist, den Benutzerzugriff auf bestimmte Objekte zu steuern, müssen Sie möglicherweise die Sicherheitsaspekte bei der Verwendung von IBM WebSphere MQ Explorer berücksichtigen.

Berechtigung zur Verwendung von IBM WebSphere MQ Explorer

Jeder Benutzer kann den IBM WebSphere MQ Explorer verwenden, allerdings sind bestimmte Berechtigungen für die Verbindung, den Zugriff und die Verwaltung von Warteschlangenmanagern erforderlich.

Zur Ausführung lokaler Verwaltungstasks mit WebSphere MQ Explorer muss ein Benutzer über die erforderliche Berechtigung zur Ausführung der Verwaltungstasks verfügen. Wenn der Benutzer Mitglied der Gruppe mqm ist, hat der Benutzer die Berechtigung, alle lokalen Verwaltungstasks auszuführen.

Um eine Verbindung zu einem fernen Warteschlangenmanager herzustellen und ferne Verwaltungstasks mit dem WebSphere MQ Explorer auszuführen, muss der Benutzer, der den WebSphere MQ Explorer ausführt, über die folgenden Berechtigungen verfügen:

- CONNECT-Berechtigung für das Ziel-WS-Manager-Objekt
- Berechtigung INQUIRE für das Ziel-WS-Manager-Objekt
- Berechtigung DISPLAY für das Ziel-WS-Manager-Objekt
- Berechtigung INQUIRE für die Warteschlange, SYSTEM.MQEXPLORER.REPLY.MODEL
- Berechtigung DISPLAY für die Warteschlange SYSTEM.MQEXPLORER.REPLY.MODEL
- Berechtigung INPUT (get) für die Warteschlange, SYSTEM.MQEXPLORER.REPLY.MODEL
- Berechtigung OUTPUT (put) für die Warteschlange, SYSTEM.ADMIN.COMMAND.QUEUE
- Berechtigung INQUIRE in der Warteschlange, SYSTEM.ADMIN.COMMAND.QUEUE
- Berechtigung zum Ausführen der ausgewählten Aktion

Anmerkung: Die INPUT-Berechtigung bezieht sich auf die Eingabe für den Benutzer aus einer Warteschlange (eine get-Operation). Die OUTPUT-Berechtigung bezieht sich auf die Ausgabe des Benutzers in eine Warteschlange (eine PUT-Operation).

Um unter WebSphere MQ for z/OS eine Verbindung zu einem fernen Warteschlangenmanager herzustellen und ferne Verwaltungstasks mit dem IBM WebSphere MQ Explorer auszuführen, müssen Sie Folgendes angeben:

- Ein RACF -Profil für die Systemwarteschlange SYSTEM.MQEXPLORER.REPLY.MODEL
- Ein RACF -Profil für die Warteschlangen, AMQ.MQEXPLORER.*

Darüber hinaus muss der Benutzer, der WebSphere MQ Explorer ausführt, über die folgenden Berechtigungen verfügen:

- RACF -Berechtigung UPDATE für die Systemwarteschlange SYSTEM.MQEXPLORER.REPLY.MODEL
- RACF -Berechtigung UPDATE für die Warteschlangen, AMQ.MQEXPLORER.*
- CONNECT-Berechtigung für das Ziel-WS-Manager-Objekt
- Berechtigung zum Ausführen der ausgewählten Aktion

- Berechtigung READ für alle hlq.DISPLAY.Objekt-Profile in der Klasse MQCMDS

Informationen zum Erteilen von Berechtigungen für WebSphere MQ -Objekte finden Sie im Abschnitt Zugriff auf ein WebSphere MQ -Objekt auf UNIX -oder Linux -Systemen und Windows erteilen.

Wenn ein Benutzer versucht, eine Operation auszuführen, die er nicht ausführen darf, ruft der Zielwarteschlangenmanager Berechtigungsfehler auf, und die Operation schlägt fehl.

Der Standardfilter im WebSphere MQ Explorer ist die Anzeige aller WebSphere MQ -Objekte. Wenn es WebSphere MQ -Objekte gibt, für die ein Benutzer keine Anzeigeberechtigung hat, werden Berechtigungsfehler generiert. Wenn Berechtigungsereignisse aufgezeichnet werden, schränken Sie den Bereich der Objekte ein, die für die Objekte angezeigt werden, für die der Benutzer die Berechtigung DISPLAY für hat.

Sicherheit für die Verbindung zu fernen Warteschlangenmanagern

Sie müssen die Kanäle zwischen IBM WebSphere MQ Explorer und jedem fernen Warteschlangenmanager schützen.

Der IBM WebSphere MQ Explorer stellt eine Verbindung zu fernen Warteschlangenmanagern als MQI-Clientanwendung her. Dies bedeutet, dass jeder ferne WS-Manager über eine Definition eines Serververbindungskanals und eines geeigneten TCP/IP-Listeners verfügen muss. Wenn Sie Ihren Serververbindungskanal nicht sichern, ist es möglich, dass eine heimtückische Anwendung eine Verbindung zum selben Serververbindungskanal herstellt und Zugriff auf die WS-Manager-Objekte mit unbegrenzter Berechtigung erhält. Um Ihren Serververbindungskanal zu sichern, geben Sie für das MCAUSER-Attribut des Kanals einen Wert an, der nicht leer ist, verwenden Sie Kanalauthentifizierungsdatensätze oder verwenden Sie einen Sicherheitsexit.

Der Standardwert für das Attribut MCAUSER ist die lokale Benutzer-ID . Wenn Sie einen nicht leeren Benutzernamen als MCAUSER-Attribut des Serververbindungskanals angeben, werden alle Programme, die mit diesem Kanal eine Verbindung zum WS-Manager herstellen, mit der Identität des benannten Benutzers ausgeführt und haben dieselbe Berechtigungsstufe. Dies ist nicht der Fall, wenn Sie Kanalauthentifizierungsdatensätze verwenden.

Sicherheitsexit mit WebSphere MQ Explorer verwenden

Mit dem WebSphere MQ Explorer können Sie einen Standardsicherheitsexit und Warteschlangenmanagerspezifische Sicherheitsexits angeben.

Sie können einen Standardsicherheitsexit definieren, der für alle neuen Clientverbindungen aus WebSphere MQ Explorer verwendet werden kann. Dieser Standardexit kann zum Zeitpunkt der Verbindungsabschlussverbindung überschrieben werden. Sie können auch einen Sicherheitsexit für einen einzelnen Warteschlangenmanager oder eine Gruppe von Warteschlangenmanagern definieren, die wirksam werden, wenn eine Verbindung hergestellt wird. Sie geben Exits mit WebSphere MQ Explorer an. Weitere Informationen finden Sie im WebSphere MQ Help Center.

Mit IBM WebSphere MQ Explorer über SSL-fähige MQI-Kanäle eine Verbindung zu einem fernen Warteschlangenmanager herstellen

Der IBM WebSphere MQ Explorer stellt über einen MQI-Kanal Verbindungen zu fernen Warteschlangenmanagern her. Wenn Sie den MQI-Kanal mit der SSL-Sicherheit sichern wollen, müssen Sie den Kanal mit Hilfe einer Definitionstabelle für Clientkanäle einrichten.

Informationen zum Einrichten eines MQI-Kanals mithilfe einer Definitionstabelle für Clientkanäle finden Sie unter Übersicht über IBM WebSphere MQ MQI-Clients.

Nachdem Sie den Kanal mithilfe einer Definitionstabelle für Clientkanäle eingerichtet haben, können Sie mit dem IBM WebSphere MQ Explorer über den SSL-fähigen MQI-Kanal eine Verbindung zu einem fernen Warteschlangenmanager herstellen, wie in „Tasks auf dem System, auf dem sich der ferne WS-Manager befindet” auf Seite 62 und „Tasks auf dem System, das die IBM WebSphere MQ Explorer hostet” auf Seite 63 beschrieben.

Tasks auf dem System, auf dem sich der ferne WS-Manager befindet

Führen Sie auf dem System, auf dem sich der ferne WS-Manager befindet, die folgenden Tasks aus:

1. Definieren Sie ein Serververbindungs-und Clientverbindungspaar, und geben Sie den entsprechenden Wert für die Variable *SSLCIPH* auf der Serververbindung auf beiden Kanälen an. Weitere Informationen zur Variablen *SSLCIPH* finden Sie im Abschnitt Kanäle mit SSL schützen.
2. Senden Sie die Kanaldefinitionstabelle `AMQCLCHL.TAB` , die sich im Verzeichnis `@ipcc` des Warteschlangenmanagers befindet, an das System, auf dem sich die IBM WebSphere MQ Explorerbefindet.
3. Starten Sie einen TCP/IP-Listener an einem angegebenen Port.
4. Stellen Sie sowohl die CA-als auch die persönlichen SSL-Zertifikate in das SSL-Verzeichnis des Warteschlangenmanagers:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` für UNIX and Linux -Systeme
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` für Windows -Systeme

   Dabei steht +QMNAME+ für ein Token, das den Namen des Warteschlangenmanagers darstellt.
5. Erstellen Sie eine Schlüsseldatenbankdatei des Typs CMS mit dem Namen `key.kdb` . Verstashen Sie das Kennwort in einer Datei, indem Sie entweder die Option in der grafischen Benutzerschnittstelle von iKeyman aktivieren oder die Option `-stash` mit den Befehlen **runmqckm** verwenden.
6. Fügen Sie die CA-Zertifikate der Schlüsseldatenbank hinzu, die im vorherigen Schritt erstellt wurde.
7. Importieren Sie das persönliche Zertifikat für den WS-Manager in die Schlüsseldatenbank.

Ausführlichere Informationen zur Arbeit mit Secure Sockets Layer auf Windows -Systemen finden Sie unter Mit SSL oder TLS auf UNIX-, Linux -und Windows -Systemen arbeiten .

Tasks auf dem System, das die IBM WebSphere MQ Explorer hostet

Führen Sie auf dem System, das den IBM WebSphere MQ Explorer hostet, folgende Tasks aus:

1. Erstellen Sie eine Schlüsseldatenbankdatei des Typs JKS mit dem Namen `key.jks`. Legen Sie ein Kennwort für diese Schlüsseldatenbankdatei fest.

   IBM WebSphere MQ Explorer verwendet für SSL-Sicherheit Java-Keystore-Dateien (JKS). Die für die Konfiguration von SSL für IBM WebSphere MQ Explorer erstellte Keystore-Datei muss daher mit der betreffenden JKS-Datei übereinstimmen.
2. Fügen Sie die CA-Zertifikate der Schlüsseldatenbank hinzu, die im vorherigen Schritt erstellt wurde.
3. Importieren Sie das persönliche Zertifikat für den WS-Manager in die Schlüsseldatenbank.
4. Starten Sie auf Windows -und Linux -Systemen MQ Explorer über das Systemmenü, die ausführbare Datei `MQExplorer` oder den Befehl **strmqcfg** .
5. Klicken Sie in der IBM WebSphere MQ Explorer -Symbolleiste auf **Fenster-> Vorgaben**, erweitern Sie dann **WebSphere MQ Explorer** und klicken auf **SSL-Client-Zertifikatsspeicher**. Geben Sie den Namen und das Kennwort für die in Schritt 1 von „Tasks auf dem System, das die IBM WebSphere MQ Explorer hostet” auf Seite 63 erstellte JKS-Datei im vertrauenswürdigen Zertifikatsspeicher und im persönlichen Zertifikatsspeicher ein, und klicken Sie anschließend auf **OK** .
6. Schließen Sie das Fenster **Benutzervorgaben** , und klicken Sie auf **Warteschlangenmanager** . Klicken Sie auf **Warteschlangenmanager anzeigen/verdecken** und anschließend in der Anzeige **Warteschlangenmanager anzeigen/verdecken** auf **Hinzufügen** .
7. Geben Sie den Namen des Warteschlangenmanagers ein, und wählen Sie die Option **Direkt verbinden** aus. Klicken Sie auf ' Weiter
8. Wählen Sie die Option **Clientkanaldefinitionstabelle (CCDT) verwenden** aus und geben Sie die Position der Kanaltabellendatei an, die Sie vom fernen Warteschlangenmanager in Schritt 2 in „Tasks auf dem System, auf dem sich der ferne WS-Manager befindet” auf Seite 62 auf dem System, auf dem sich der ferne WS-Manager befindet, übertragen haben.
9. Klicken Sie auf **Fertig stellen** . Sie können jetzt über IBM WebSphere MQ Explorerauf den fernen Warteschlangenmanager zugreifen.

Verbindung über einen anderen Warteschlangenmanager herstellen

Mit dem IBM WebSphere MQ Explorer können Sie über einen temporären Warteschlangenmanager, mit dem der IBM WebSphere MQ bereits verbunden ist, eine Verbindung zu einem Warteschlangenmanager herstellen.

In diesem Fall stellt der IBM WebSphere MQ Explorer die PCF-Befehlsnachrichten in den temporären Warteschlangenmanager und gibt Folgendes an:

- Den Parameter *ObjectQMgrName* im Objektdeskriptor (MQOD) als Name des Zielwarteschlangenmanagers. Weitere Informationen zur Auflösung von Warteschlangennamen finden Sie unter Namensauflösung .
- Den Parameter *UserIdentifier* im Nachrichtendeskriptor (MQMD) als lokale Benutzer-ID (userId).

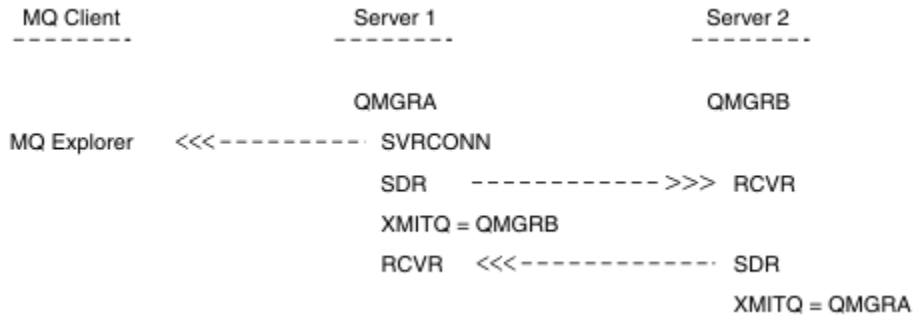
Wenn die Verbindung dann zum Herstellen der Verbindung zum Zielwarteschlangenmanager über einen temporären Warteschlangenmanager verwendet wird, wird die Benutzer-ID erneut im Parameter *UserIdentifier* des Nachrichtendeskriptors (MQMD) ausgegeben. Damit der MCA-Listener auf dem Zielwarteschlangenmanager diese Nachricht akzeptieren kann, muss entweder das Attribut MCAUSER festgelegt werden, oder die Benutzer-ID muss bereits mit der Berechtigung put vorhanden sein.

Der Befehlsserver auf dem Zielwarteschlangenmanager reiht Nachrichten in die Übertragungswarteschlange ein, die die userId im Parameter *UserIdentifier* im Nachrichtendeskriptor (MQMD) angibt. Damit dies erfolgreich ist, muss die userId bereits mit der Berechtigung 'put' auf dem Zielwarteschlangenmanager vorhanden sein.

Das folgende Beispiel zeigt, wie Sie einen Warteschlangenmanager über einen temporären Warteschlangenmanager mit WebSphere MQ Explorer verbinden.

Stellen Sie eine Fernverwaltungsverbindung zu einem Warteschlangenmanager her. Überprüfen Sie Folgendes:

- Der WS-Manager auf dem Server ist aktiv und hat einen Serververbindungskanal (SVRCONN) definiert.
- Listener ist aktiv.
- Der Befehlsserver ist aktiv.
- Die Warteschlange SYSTEM.MQ EXPLORER.REPLY.MODEL wurde erstellt, und Sie verfügen über ausreichende Berechtigungen.
- WS-Manager-Listener, Befehlsserver und Senderkanäle werden gestartet.



In diesem Beispiel gilt Folgendes:

- IBM WebSphere MQ Explorer ist über eine Clientverbindung mit Warteschlangenmanager QMGRA (auf Server1) verbunden.
- Warteschlangenmanager QMGRB auf Server2 kann jetzt über einen temporären Warteschlangenmanager (QMGRA) mit IBM WebSphere MQ Explorer verbunden werden
- Wenn Sie mit WebSphere MQ Explorer eine Verbindung zu QMGRB herstellen, wählen Sie QMGRA als temporären Warteschlangenmanager aus.

In dieser Situation gibt es keine direkte Verbindung zu QMGRB aus dem IBM WebSphere MQ Explorer; die Verbindung zu QMGRB erfolgt über QMGRA.

Der WS-Manager QMGRB auf Server2 ist mit QMGRA auf Server1 mit Sender-Empfänger-Kanälen verbunden. Der Kanal zwischen QMGRA und QMGRB muss so konfiguriert werden, dass eine Fernverwaltung möglich ist (siehe „Kanäle und Übertragungswarteschlangen für die Fernverwaltung vorbereiten” auf Seite 112).

Warteschlangenmanager und Cluster anzeigen und ausblenden

In IBM WebSphere MQ Explorer können mehrere Warteschlangenmanager gleichzeitig angezeigt werden. In der Anzeige 'Warteschlangenmanager ein-/ausblenden' können Sie auswählen, ob Sie Informationen zu einer anderen (fernen) Maschine anzeigen können (im Menü können Sie die Option für den Warteschlangenmanagerknoten auswählen). Lokale WS-Manager werden automatisch erkannt.

So zeigen Sie einen fernen Warteschlangenmanager an:

1. Klicken Sie mit der rechten Maustaste auf den Baumknoten `Queue Managers` und wählen Sie dann *Warteschlangenmanager anzeigen/verdecken ...*aus.
2. Klicken Sie auf **Hinzufügen**. Die Anzeige 'WS-Manager anzeigen/verdecken' wird angezeigt.
3. Geben Sie den Namen des fernen Warteschlangenmanagers und den Hostnamen oder die IP-Adresse in die angegebenen Felder ein.

   Der Hostname oder die IP-Adresse wird verwendet, um eine Clientverbindung zum fernen Warteschlangenmanager mit dem Standard-Serververbindungskanal SYSTEM.ADMIN.SVRCONN oder einem benutzerdefinierten Serververbindungskanal herzustellen.
4. Klicken Sie auf **Fertigstellen**.

In der Anzeige 'WS-Manager anzeigen/verdecken' wird auch eine Liste aller sichtbaren Warteschlangenmanager angezeigt. Sie können diese Anzeige verwenden, um Warteschlangenmanager in der Navigationssicht zu verdecken.

Wenn IBM WebSphere MQ Explorer einen Warteschlangenmanager anzeigt, der Mitglied eines Clusters ist, wird der Cluster erkannt und automatisch angezeigt.

Gehen Sie wie folgt vor, um die Liste der fernen WS-Manager aus dieser Anzeige zu exportieren:

1. Schließen Sie die Anzeige 'WS-Manager anzeigen/verdecken'.
2. Klicken Sie mit der rechten Maustaste auf den obersten Knoten der Baumstruktur **IBM WebSphere MQ** im Navigationsfenster von WebSphere MQ Explorer und wählen Sie dann **MQ Explorer-Einstellungen exportieren** aus.
3. Klicken Sie auf **MQ Explorer > MQ Explorer-Einstellungen** .
4. Wählen Sie **Verbindungsdaten > Ferne Warteschlangenmanager** aus.
5. Wählen Sie eine Datei aus, in der die exportierten Einstellungen gespeichert werden sollen.
6. Klicken Sie abschließend auf **Fertig stellen** , um die Verbindungsinformationen des fernen Warteschlangenmanagers in die angegebene Datei zu exportieren.

Gehen Sie wie folgt vor, um eine Liste der fernen Warteschlangenmanager zu importieren

1. Klicken Sie mit der rechten Maustaste auf den obersten Knoten der Baumstruktur **IBM WebSphere MQ** im Navigationsfenster von WebSphere MQ Explorer und wählen Sie dann **MQ Explorer-Einstellungen importieren** aus.
2. Klicken Sie auf **MQ Explorer > MQ Explorer-Einstellungen** .
3. Klicken Sie auf **Durchsuchen** und navigieren Sie zu dem Pfad der Datei, die die Verbindungsinformationen des fernen Warteschlangenmanagers enthält.
4. Klicken Sie auf **Öffnen** . Wenn die Datei eine Liste mit fernen Warteschlangenmanagern enthält, ist das Kästchen **Verbindungsdaten > Ferne Warteschlangenmanager** aktiviert.
5. Klicken Sie abschließend auf **Fertigstellen** , um die Verbindungsinformationen für den fernen Warteschlangenmanager in WebSphere MQ Explorer zu importieren.

Clusterzugehörigkeit

IBM WebSphere MQ Explorer benötigt Informationen zu Warteschlangenmanagern, die Mitglied eines Clusters sind.

Wenn ein Warteschlangenmanager Mitglied eines Clusters ist, wird der Clusterknoten automatisch gefüllt.

Wenn Warteschlangenmanager zu Mitgliedern von Clustern werden, während der IBM WebSphere MQ Explorer ausgeführt wird, müssen Sie den IBM WebSphere MQ Explorer mit aktuellen Verwaltungsdaten zu Clustern versorgen, damit er effizient mit ihnen kommunizieren und bei einer entsprechenden Anforderung die richtigen Clusterinformationen anzeigen kann. Dazu benötigt der WebSphere MQ Explorer die folgenden Informationen:

- Name eines Repository-Warteschlangenmanagers
- Verbindungsname des Repository-Warteschlangenmanagers, wenn er sich auf einem fernen Warteschlangenmanager befindet

Mit diesen Informationen kann WebSphere MQ Explorer Folgendes ausführen:

- Den Repository-Warteschlangenmanager verwenden, um eine Liste der Warteschlangenmanager im Cluster abzurufen.
- Die Warteschlangenmanager verwalten, die Mitglieder des Clusters sind und sich auf unterstützten Plattformen und Befehlsebenen befinden.

Eine Verwaltung ist nicht möglich, wenn:

- Das ausgewählte Repository ist nicht verfügbar. WebSphere MQ Explorer wechselt nicht automatisch zu einem alternativen Repository.
- Das ausgewählte Repository kann nicht über TCP/IP kontaktiert werden.
- Das ausgewählte Repository wird auf einem Warteschlangenmanager ausgeführt, der auf einer Plattform und Befehlsebene ausgeführt wird, die von WebSphere MQ Explorer nicht unterstützt werden.

Die Clustermitglieder, die verwaltet werden können, können lokal sein, oder sie können fern sein, wenn sie über TCP/IP kontaktiert werden können. Der IBM WebSphere MQ Explorer stellt zu lokalen Warteschlangenmanagern, die Mitglieder eines Clusters sind, direkte Verbindungen her, ohne eine Clientverbindung zu verwenden.

Datenkonvertierung

Der IBM WebSphere MQ Explorer arbeitet in CCSID 1208 (UTF-8). Dies ermöglicht es dem IBM WebSphere MQ Explorer, die Daten von fernen Warteschlangenmanagern richtig anzuzeigen. Unabhängig davon, ob die Verbindung zu einem Warteschlangenmanager direkt oder über einen temporären Warteschlangenmanager hergestellt wird, müssen für den IBM WebSphere MQ Explorer alle ankommenden Nachrichten nach CCSID 1208 (UTF-8) konvertiert werden.

Es wird eine Fehlernachricht ausgegeben, wenn Sie versuchen, eine Verbindung zwischen dem IBM WebSphere MQ Explorer und einem Warteschlangenmanager mit einer CCSID herzustellen, die der IBM WebSphere MQ Explorernicht erkennt.

Unterstützte Konvertierungen werden im Abschnitt Codepagekonvertierung beschrieben.

Sicherheit unter Windows

Der Assistent WebSphere MQ erstellt ein spezielles Benutzerkonto, damit der Windows -Service von Prozessen gemeinsam genutzt werden kann, die ihn verwenden müssen.

Ein Windows -Service wird von Clientprozessen für eine IBM WebSphere MQ -Installation gemeinsam genutzt. Für jede Installation wird ein Service erstellt. Jeder Service hat den Namen `MQ_`*`Installation-Name`*und den Anzeigenamen `IBM WebSphere MQ(`*`InstallationName`*`)`. Before Version 7.1, with only one installation on a server the single, Windows service was named `MQSeriesServices` with the display name `IBM MQSeries`.

Da jeder Service von nicht interaktiven und interaktiven Anmeldesitzungen gemeinsam genutzt werden muss, müssen Sie jeden Service unter einem speziellen Benutzeraccount starten. Sie können ein spezielles Benutzerkonto für alle Services verwenden oder verschiedene spezielle Benutzerkonten erstellen. Jedes spezielle Benutzerkonto muss über die Benutzerberechtigung "Anmelden als Dienst" verfügen. Weitere Informationen finden Sie unter „Für einen IBM WebSphere MQ Windows -Service erforderliche Benutzerberechtigungen” auf Seite 68. Wenn die Benutzer-ID nicht über die Berechtigung zur Ausführung des Service verfügt, kann der Service nicht gestartet werden und im Windows-Systemereignisprotokoll wird ein Fehler gemeldet. Normalerweise müssen Sie den IBM WebSphere MQ-Vorbereitungsassistenten ausführen, um die Benutzer-ID korrekt einzurichten. Wenn Sie die Benutzer-ID jedoch manuell konfiguriert haben, ist es möglich, dass Sie ein Problem haben, das Sie beheben müssen.

Wenn Sie IBM WebSphere MQ installieren und den IBM WebSphere MQ -Vorbereitungsassistenten zum ersten Mal ausführen, erstellt er ein lokales Benutzerkonto für den Service mit dem Namen MUSR_MQADMIN mit den erforderlichen Einstellungen und Berechtigungen, einschließlich "Anmelden als Dienst".

Für nachfolgende Installationen erstellt der Vorbereitungsassistent IBM WebSphere MQ ein Benutzerkonto mit dem Namen MUSR_MQADMINx, wobei x für die nächste verfügbare Zahl steht, die eine Benutzer-ID darstellt, die nicht vorhanden ist. Das Kennwort für MUSR_MQADMINx wird nach dem Zufallsprinzip generiert, wenn das Konto erstellt wird und die Anmeldeumgebung für den Service konfiguriert wird. Das generierte Kennwort läuft nicht ab.

Dieses IBM WebSphere MQ-Konto wird nicht von Kontorichtlinien beeinträchtigt, die im System eingerichtet sind und durch die Kennwörter für das Konto nach einem bestimmten Zeitraum geändert werden müssen.

Das Kennwort ist außerhalb dieser einmaligen Verarbeitung nicht bekannt und wird vom Windows-Betriebssystem in einem sicheren Teil der Registry gespeichert.

Active Directory verwenden (nurWindows)

In einigen Netzkonfigurationen, in denen Benutzerkonten auf Domänencontrollern definiert sind, die Active Directoryverwenden, verfügt das lokale Benutzerkonto, unter dem IBM WebSphere MQ ausgeführt wird, möglicherweise nicht über die Berechtigung, die zum Abfragen der Gruppenzugehörigkeit anderer Domänenbenutzerkonten erforderlich ist. Der IBM WebSphere MQ -Vorbereitungsassistent gibt an, ob dies der Fall ist, indem Tests durchgeführt und die Benutzer Fragen zur Netzkonfiguration gestellt werden.

Wenn das lokale Benutzerkonto IBM WebSphere MQ nicht über die erforderliche Berechtigung verfügt, fordert der IBM WebSphere MQ -Vorbereitungsassistent den Benutzer zur Eingabe der Kontodetails eines Domänenbenutzerkontos mit bestimmten Benutzerberechtigungen auf. Informationen zu den Benutzerberechtigungen, die das Domänenbenutzerkonto erfordert, finden Sie in „Für einen IBM WebSphere MQ Windows -Service erforderliche Benutzerberechtigungen” auf Seite 68. Nachdem der Benutzer gültige Kontodetails für das Domänenbenutzerkonto in den Vorbereitungsassistenten für IBM WebSphere MQ eingegeben hat, konfiguriert er einen IBM WebSphere MQ Windows -Service, der unter dem neuen Account ausgeführt werden soll. Die Kontodetails werden im sicheren Teil der Registry festgehalten und können nicht von den Benutzern gelesen werden.

Wenn der Service aktiv ist, wird ein IBM WebSphere MQ Windows -Service gestartet und bleibt so lange aktiv, wie der Service aktiv ist. Ein IBM WebSphere MQ -Administrator, der sich am Server anmeldet, nachdem der Windows -Service gestartet wurde, kann die IBM WebSphere MQ Explorer zum Verwalten von Warteschlangenmanagern auf dem Server verwenden. Dadurch wird die IBM WebSphere MQ Explorer mit dem vorhandenen Windows -Serviceprozess verbunden. Diese beiden Aktionen benötigen unterschiedliche Berechtigungsstufen, bevor sie funktionieren können:

- Für den Startprozess ist eine Startberechtigung erforderlich.
- Der IBM WebSphere MQ-Administrator benötigt eine Zugriffsberechtigung.

Für einen IBM WebSphere MQ Windows -Service erforderliche Benutzerberechtigungen

In der Tabelle in diesem Abschnitt sind die Benutzerberechtigungen aufgelistet, die für das lokale Benutzerkonto und das Domänenbenutzerkonto erforderlich sind, unter dem der Windows -Service für eine IBM WebSphere MQ -Installation ausgeführt wird.

Als Stapeljob anmelden	Aktiviert die Ausführung eines IBM WebSphere MQ Windows -Service unter diesem Benutzerkonto.
Als Dienst anmelden	Ermöglicht Benutzern, den IBM WebSphere MQ Windows -Service für die Anmeldung mit dem konfigurierten Konto festzulegen.
System herunterfahren	Ermöglicht dem IBM WebSphere MQ Windows -Service, den Server erneut zu starten, wenn dies entsprechend konfiguriert ist, wenn die Wiederherstellung eines Service fehlschlägt.
Kontingente erhöhen	Erforderlich für den Betriebssystemaufruf `CreateProcessAsUser` .
Einsetzen als Teil des Betriebssystems	Erforderlich für den Betriebssystemaufruf `LogonUser` .
Durchgangsprüfung umgehen	Erforderlich für den Betriebssystemaufruf `LogonUser` .
Ersetzen Sie ein Token auf Prozessebene.	Erforderlich für den Betriebssystemaufruf `LogonUser` .

Anmerkung: In Umgebungen, in der ASP-und IIS-Anwendungen ausgeführt werden, sind möglicherweise Debugprogrammberechtigungen erforderlich.

Für Ihr Domänenbenutzerkonto müssen diese Windows -Benutzerberechtigungen als effektive Benutzerberechtigungen festgelegt sein, die in der Anwendung "Lokale Sicherheitsrichtlinie" aufgelistet sind. Ist dies nicht der Fall, setzen Sie sie entweder lokal auf dem Server mit der Anwendung "Lokale Sicherheitsrichtlinie" oder in der Domäne "Domäne Security Application" (Domänensicherheitsanwendung).

Dem IBM WebSphere MQ -Service zugeordneten Benutzernamen ändern

Möglicherweise müssen Sie den Benutzernamen, der dem IBM WebSphere MQ -Service zugeordnet ist, von MUSR_MQADMIN in einen anderen Namen ändern. (Dies kann beispielsweise erforderlich sein, wenn Ihr WS-Manager DB2zugeordnet ist, das keine Benutzernamen mit mehr als 8 Zeichen akzeptiert.)

Vorgehensweise

1. Erstellen Sie einen neuen Benutzeraccount (z. B. **NEW_NAME**).
2. Verwenden Sie den IBM WebSphere MQ -Vorbereitungsassistenten, um die Details des neuen Benutzeraccounts einzugeben.

Kennwort des IBM WebSphere MQ Windows -Servicebenutzeraccounts ändern

Informationen zu diesem Vorgang

Führen Sie die folgenden Schritte aus, um das Kennwort des lokalen Benutzeraccounts für denIBM WebSphere MQ Windows -Service zu ändern:

Vorgehensweise

1. Geben Sie den Benutzer an, unter dem der Service ausgeführt wird.
2. Stoppen Sie den IBMIBM WebSphere MQ -Service über die Computerverwaltungsanzeige.

3. Ändern Sie das erforderliche Kennwort auf die gleiche Weise wie das Kennwort einer Person.
4. Rufen Sie in der Anzeige 'Computerverwaltung' die Eigenschaften für den IBM WebSphere MQ -Service auf.
5. Wählen Sie die Seite **Log On** aus.
6. Bestätigen Sie, dass der angegebene Accountname mit dem Benutzer übereinstimmt, für den das Kennwort geändert wurde.
7. Geben Sie das Kennwort in die Felder **Kennwort** und **Kennwort bestätigen** ein, und klicken Sie auf **OK** .

IBM WebSphere MQ Windows -Service für eine Installation, die unter einem Domänenbenutzerkonto ausgeführt wird

Informationen zu diesem Vorgang

Wenn der IBM WebSphere MQ Windows -Service für eine Installation unter einem Domänenbenutzerkonto ausgeführt wird, können Sie das Kennwort für das Konto wie folgt ändern:

Vorgehensweise

1. Ändern Sie das Kennwort für das Domänenkonto auf dem Domänencontroller. Möglicherweise müssen Sie Ihren Domänenadministrator bitten, dies für Sie zu tun.
2. Führen Sie die Schritte zum Ändern der Seite **Anmelden** für den IBMIBM WebSphere MQ -Service aus.

   Das Benutzerkonto, unter dem der IBM WebSphere MQ Windows -Service ausgeführt wird, führt alle MQSC-Befehle aus, die von Benutzerschnittstellenanwendungen ausgegeben oder beim Systemstart, Systemabschluss oder bei der Servicewiederherstellung automatisch ausgeführt werden. Dieses Benutzerkonto muss deshalb über Administratorberechtigungen für IBM WebSphere MQ verfügen. Standardmäßig wird sie der lokalen Gruppe **mqm** auf dem Server hinzugefügt. Wenn diese Mitgliedschaft entfernt wird, funktioniert der IBM WebSphere MQ Windows -Service nicht. Weitere Informationen zu Benutzerberechtigungen finden Sie in „Für einen IBM WebSphere MQ Windows -Service erforderliche Benutzerberechtigungen” auf Seite 68.

   Wenn ein Sicherheitsproblem mit dem Benutzerkonto auftritt, unter dem der IBM WebSphere MQ Windows -Service ausgeführt wird, werden Fehlernachrichten und Beschreibungen im Systemereignisprotokoll angezeigt.

Zugehörige Konzepte

„Active Directory verwenden (nurWindows)” auf Seite 67

In einigen Netzkonfigurationen, in denen Benutzerkonten auf Domänencontrollern definiert sind, die Active Directoryverwenden, verfügt das lokale Benutzerkonto, unter dem IBM WebSphere MQ ausgeführt wird, möglicherweise nicht über die Berechtigung, die zum Abfragen der Gruppenzugehörigkeit anderer Domänenbenutzerkonten erforderlich ist. Der IBM WebSphere MQ -Vorbereitungsassistent gibt an, ob dies der Fall ist, indem Tests durchgeführt und die Benutzer Fragen zur Netzkonfiguration gestellt werden.

IBM WebSphere MQ Koordination mit Db2 als Ressourcenmanager

Wenn Sie Ihre Warteschlangenmanager über die IBM WebSphere MQ Explorer starten oder IBM WebSphere MQ V7 verwenden und Probleme bei der Koordination von Db2 haben, überprüfen Sie die Fehlerprotokolle des Warteschlangenmanagers.

Suchen Sie in den Fehlerprotokollen Ihres Warteschlangenmanagers nach einem Fehler wie dem folgenden:

```
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.
```

Erläuterung: Die Benutzer-ID (der Standardname ist MUSR_MQADMIN), die den IBM WebSphere MQ-Serviceprozess `amqsvc.exe` ausführt, verwendet immer noch ein Zugriffstoken, das keine Gruppenzugehörigkeitsinformationen für die Gruppe DB2USERS enthält.

Solve: Nachdem Sie sichergestellt haben, dass die IBM WebSphere MQ -Servicebenutzer-ID ein Mitglied der DB2USERS-Instanz ist, verwenden Sie die folgende Befehlsfolge:

- Stoppen Sie den Service.
- alle anderen Prozesse stoppen, die unter derselben Benutzer-ID ausgeführt werden.
- starten Sie diese Prozesse erneut.

Ein Warmstart des Systems würde die genannten Schritte sicherstellen, ist aber nicht notwendig.

IBM WebSphere MQ Explorer erweitern

IBM WebSphere MQ für Windowsund IBM WebSphere MQ für Linux (Plattformenx86 und x86-64) stellen eine Verwaltungsschnittstelle mit dem Namen IBM WebSphere MQ Explorer bereit, um Verwaltungstasks als Alternative zur Verwendung von Steuer-oder MQSC-Befehlen auszuführen.

Diese Informationen gelten nur für WebSphere MQ für Windows und WebSphere MQ für Linux (x86 -und x86-64 -Plattformen).

Der IBM WebSphere MQ Explorer stellt Informationen in einem Stil dar, der mit dem des Eclipse -Frameworks und den anderen Plug-in-Anwendungen, die Eclipse unterstützt, konsistent ist.

Durch die Erweiterung von IBM WebSphere MQ Explorer können Systemadministratoren den WebSphere MQ Explorer anpassen, um die Verwaltung von WebSphere MQzu verbessern.

Weitere Informationen finden Sie unter *IBM WebSphere MQ Explorer erweitern* in der Produktdokumentation zu IBM WebSphere MQ Explorer.

IBM WebSphere MQ -Taskleistenanwendung verwenden (nur Windows)

Die IBM WebSphere MQ -Taskleistenanwendung zeigt ein Symbol in der Windows -Taskleiste auf dem Server an. Das Symbol gibt den aktuellen Status von IBM WebSphere MQ an und stellt ein Menü bereit, aus dem Sie einfache Aktionen ausführen können.

Unter Windowsbefindet sich das Symbol WebSphere MQ in der Taskleiste des Servers und wird mit einem farbigen Statussymbol überlagert, das eine der folgenden Bedeutungen haben kann:

Grün
Ordnungsgemäßer Betrieb; keine Alerts vorhanden.

Blau
Unbestimmt; WebSphere MQ wird gestartet oder heruntergefahren

Gelb
Alert: Ein oder mehrere Services sind fehlgeschlagen oder sind bereits fehlgeschlagen.

Klicken Sie mit der rechten Maustaste auf das Symbol WebSphere MQ , um das Menü anzuzeigen. Über das Menü können Sie folgende Aktionen ausführen:

- Klicken Sie auf **Öffnen** , um den Alertmonitor WebSphere MQ zu öffnen.
- Klicken Sie auf **Beenden** , um die WebSphere MQ -Taskleistenanwendung zu beenden
- Klicken Sie auf **WebSphere MQ Explorer** , um IBM WebSphere MQ Explorer zu starten.
- Klicken Sie auf **WebSphere MQ** , um WebSphere MQ zu stoppen.
- Klicken Sie auf **About WebSphere MQ** , um Informationen zum WebSphere MQ -Alertmonitor anzuzeigen.

IBM WebSphere MQ -Alertmonitoranwendung (nur Windows)

Der IBM WebSphere MQ-Alertmonitor ist ein Fehlererkennungstool, das Probleme mit IBM WebSphere MQ auf einer lokalen Maschine identifiziert und erfasst.

Der Alertmonitor zeigt Informationen zum aktuellen Status der lokalen Installation eines WebSphere MQ -Servers an. Er überwacht außerdem die Windows Advanced Configuration and Power Interface (ACPI) und stellt sicher, dass die ACPI-Einstellungen durchgesetzt werden.

Über den Alertmonitor WebSphere MQ können Sie Folgendes ausführen:

- Greifen Sie direkt auf IBM WebSphere MQ Explorer zu
- Anzeigen von Informationen zu allen ausstehenden Alerts
- Beenden Sie den WebSphere MQ -Service auf der lokalen Maschine.
- Weiterleiten von Alertnachrichten über das Netz an ein konfigurierbares Benutzerkonto oder an eine Windows -Workstation oder einen Server

Lokale IBM WebSphere MQ -Objekte verwalten

In diesem Abschnitt erfahren Sie, wie Sie lokale IBM WebSphere MQ-Objekte verwalten können, um Anwendungsprogramme zu unterstützen, die die Schnittstelle für Nachrichtenwarteschlangen (Message Queue Interface, MQI) verwenden. In diesem Kontext bedeutet lokale Verwaltung das Erstellen, Anzeigen, Ändern, Kopieren und Löschen von IBM WebSphere MQ-Objekten.

Zusätzlich zu den in diesem Abschnitt beschriebenen Methoden können Sie den IBM WebSphere MQ Explorer verwenden, um lokale WebSphere MQ -Objekte zu verwalten (siehe „Verwaltung mit IBM WebSphere MQ Explorer” auf Seite 58).

Dieser Abschnitt enthält folgende Informationen:

Warteschlangenmanager starten und stoppen

Verwenden Sie dieses Thema als Einführung zum Stoppen und Starten eines Warteschlangenmanagers.

WS-Manager starten

Verwenden Sie zum Starten eines Warteschlangenmanagers den Befehl `strmqm` folgendermaßen:

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

Auf WebSphere MQ für Windows und WebSphere MQ für Linux (x86 -und x86-64 -Plattformen) können Sie einen Warteschlangenmanager wie folgt starten:

1. Öffnen Sie den IBM WebSphere MQ Explorer.
2. Wählen Sie den Warteschlangenmanager in der Navigatoransicht aus.
3. Klicken Sie auf `Start`. Der WS-Manager wird gestartet.

Wenn der Start des Warteschlangenmanagers länger als ein paar Sekunden dauert, gibt WebSphere MQ Informationsnachrichten aus, die den Startfortschritt sporadisch detailliert beschreiben.

Der Befehl `strmqm` gibt die Steuerung erst zurück, wenn der WS-Manager gestartet wurde und bereit ist, Verbindungsanforderungen zu akzeptieren.

Warteschlangenmanager automatisch starten

In WebSphere MQ for Windows können Sie einen Warteschlangenmanager automatisch starten, wenn das System mit dem WebSphere MQ Explorer gestartet wird. Weitere Informationen finden Sie unter „Verwaltung mit IBM WebSphere MQ Explorer” auf Seite 58.

Stoppen eines Warteschlangenmanagers

Mit dem Befehl **endmqm** können Sie einen Warteschlangenmanager stoppen.

Anmerkung: Sie müssen den Befehl **endmqm** aus der Installation verwenden, die dem Warteschlangenmanager zugeordnet ist, mit dem Sie arbeiten. Um herauszufinden, welcher Installation ein Warteschlangenmanager zugeordnet ist, verwenden Sie den Befehl `dspmq -o installation`.

Soll beispielsweise ein Warteschlangenmanager mit dem Namen `QMB` gestoppt werden, geben Sie folgenden Befehl ein:

```
endmqm QMB
```

Auf WebSphere MQ für Windows und WebSphere MQ für Linux (x86 -und x86-64 -Plattformen) können Sie einen Warteschlangenmanager wie folgt stoppen:

1. Öffnen Sie den IBM WebSphere MQ Explorer.
2. Wählen Sie den Warteschlangenmanager in der Navigatoransicht aus.
3. Klicken Sie auf `Stop...`. Die Anzeige 'End Queue Manager' wird angezeigt.
4. Wählen Sie 'Gesteuert' oder 'Sofort' aus.
5. Klicken Sie auf OK. Der Warteschlangenmanager wird gestoppt.

Gesteuerter Abschluss

Standardmäßig führt der Befehl **endmqm** einen gesteuerten Abschluss des angegebenen Warteschlangenmanagers durch. Dies kann etwas länger dauern. Bei einem gesteuerten Abschluss wird gewartet, bis alle verbundenen Anwendungen getrennt sind.

Verwenden Sie diese Art der Beendigung, um Anwendungen darüber zu benachrichtigen, dass sie gestoppt werden müssen. Bei Ausgabe des Befehls:

```
endmqm -c QMB
```

erhalten Sie keine Benachrichtigung, wenn alle Anwendungen gestoppt sind. (Ein `endmqm -c QMB` -Befehl entspricht einem `endmqm QMB` -Befehl.)

Bei Ausgabe des Befehls:

```
endmqm -w QMB
```

wartet der Befehl hingegen, bis alle Anwendungen gestoppt sind und der Warteschlangenmanager beendet ist.

Sofortige Beendigung

Bei einer sofortigen Beendigung werden alle aktuellen MQI-Aufrufe zu Ende geführt, neue Aufrufe schlagen jedoch fehl. Dieser Typ des Herunterfahrens wartet nicht darauf, dass Anwendungen vom WS-Manager getrennt werden.

Geben Sie für eine sofortige Beendigung Folgendes ein:

```
endmqm -i QMB
```

Präventiver Systemabschluss

Anmerkung: Diese Abschlussmethode sollten Sie nur verwenden, wenn alle anderen Versuche, den Warteschlangenmanager mit dem Befehl **endmqm** zu stoppen, fehlgeschlagen sind. Die Auswirkungen dieser Methode auf die verbundenen Anwendungen sind unvorhersehbar.

Falls eine sofortige Beendigung nicht funktioniert, müssen Sie mit dem Flag `-p` auf einen *präventiven* Abschluss zurückgreifen. Beispiel:

```
endmqm -p QMB
```

Dadurch wird der Warteschlangenmanager sofort gestoppt. Wenn auch dies erfolglos bleibt, finden Sie unter „Manuelles Stoppen eines Warteschlangenmanagers” auf Seite 73 weitere Lösungsvorschläge.

Eine ausführliche Beschreibung des Befehls **endmqm** und seiner Optionen finden Sie im Abschnitt endmqm.

Probleme beim Abschluss eines Warteschlangenmanagers

Probleme beim Abschluss eines Warteschlangenmanagers haben häufig ihre Ursache in einzelnen Anwendungen. Zum Beispiel, wenn Anwendungen:

- MQI-Rückkehrcodes nicht korrekt überprüfen
- Keine Benachrichtigung über Quiesce anfordern
- Beenden, ohne die Verbindung zum Warteschlangenmanager zu trennen (durch Ausgabe eines `MQDISC` -Aufrufs)

Bei Problemen beim Stoppen eines Warteschlangenmanagers können Sie den Befehl **endmqm** mit der Tastenkombination STRG-C abbrechen. Danach können Sie einen anderen Befehl **endmqm** mit einem Flag für die erforderliche Abschlussmethode ausgeben.

Manuelles Stoppen eines Warteschlangenmanagers

Wenn die Standardmethoden zum Stoppen von Warteschlangenmanagern fehlschlagen, verwenden Sie die hier beschriebenen Methoden.

Die Standardmethode zum Stoppen von Warteschlangenmanagern ist die Verwendung des Befehls `endmqm`. Zum manuellen Stoppen eines Warteschlangenmanagers verwenden Sie eine der in diesem Abschnitt beschriebenen Methoden. Details zur Ausführung von Operationen für Warteschlangenmanager mit Steuerbefehlen finden Sie im Abschnitt Warteschlangenmanager erstellen und verwalten.

Warteschlangenmanager unter Windows stoppen

Vorgehensweise zum Beenden der Prozesse und des IBM WebSphere MQ -Service zum Stoppen von Warteschlangenmanagern in IBM WebSphere MQ for Windows.

So stoppen Sie einen Warteschlangenmanager, der unter WebSphere MQ für Windowsausgeführt wird:

1. Listen Sie die Namen (IDs) der aktiven Prozesse mithilfe des Windows -Task-Managers auf.
2. Beenden Sie die Prozesse mit Windows Task Manager oder dem Befehl **taskkill** in der folgenden Reihenfolge (falls sie aktiv sind):

AMQZMUC0	Kritischer Prozessmanager
AMQZXMA0	Ausführungscontroller
AMQZFUMA	OAM-Prozess
AMQZLAA0	LQM-Agenten

AMQZLSA0	LQM-Agenten
AMQZMUF0	Utility-Manager
AMQZMGR0	Prozesscontroller
AMQZMUR0	Wiederanlauffähiger Prozessmanager
AMQFQPUB	Publish/Subscribe-Prozess
AMQFCXBA	Brokerworker-Prozess
AMQRMPPA	Prozess-Pooling-Prozess
AMQCRSTA	Nicht-Thread-Responder-Jobprozess
AMQCRS6B	LU62 -Empfängerkanal und Clientverbindung
AMQRRMFA	Der Repositoryprozess (für Cluster)
AMQZDMAA	Verzögerter Nachrichtenprozessor
AMQPCSEA	Befehlsserver
RUNMQTRM	Auslösermonitor für einen Server aufrufen
RUNMQDLQ	Handler für dead-letter-Warteschlangen aufrufen
RUNMQCHI	Der Kanalinitiatorprozess
RUNMQLSR	Der Kanal-Listener-Prozess
AMQXSSVN	Gemeinsam genutzte Speicherserver
AMQZTRCN	Verfolgen

3. Stoppen Sie den WebSphere MQ -Service über die **-Verwaltungstools** > **Dienste** in der Windows -Systemsteuerung.
4. Wenn Sie alle Methoden ausprobiert haben und der WS-Manager nicht gestoppt wurde, führen Sie einen Warmstart Ihres Systems durch.

Der Windows -Task-Manager und der Befehl **tasklist** liefern begrenzte Informationen zu Tasks. Weitere Informationen zur Bestimmung der Prozesse, die sich auf einen bestimmten Warteschlangenmanager beziehen, erhalten Sie, wenn Sie ein Tool wie *Process Explorer* (procexp.exe) verwenden, das von der Microsoft-Website unter https://www.microsoft.comheruntergeladen werden kann.

Warteschlangenmanager auf UNIX and Linux -Systemen stoppen

Vorgehensweise zum Beenden der Prozesse und des IBM WebSphere MQ -Service zum Stoppen von Warteschlangenmanagern in IBM WebSphere MQ for UNIX and Linux. Sie können die hier beschriebenen Methoden ausprobieren, wenn die Standardmethoden zum Stoppen und Entfernen von Warteschlangenmanagern fehlschlagen.

Gehen Sie wie folgt vor, um einen Warteschlangenmanager zu stoppen, der unter WebSphere MQ für UNIX and Linux -Systeme ausgeführt wird:

1. Suchen Sie die Prozess-IDs der Warteschlangenmanagerprogramme, die noch mit dem Befehl ps ausgeführt werden. Wenn der WS-Manager beispielsweise QMNAME genannt wird, verwenden Sie den folgenden Befehl:

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

2. Beenden Sie alle WS-Manager-Prozesse, die noch aktiv sind. Verwenden Sie den Befehl kill , um die Prozess-IDs anzugeben, die mit dem Befehl ps erkannt wurden.

   Beenden Sie die Prozesse in der folgenden Reihenfolge:

amqzmuc0	Kritischer Prozessmanager

amqzxma0	Ausführungscontroller
amqzfuma	OAM-Prozess
amqzlaa0	LQM-Agenten
amqzlsa0	LQM-Agenten
amqzmuf0	Utility-Manager
amqzmur0	Wiederanlauffähiger Prozessmanager
amqzmgr0	Prozesscontroller
amqfqpub	Publish/Subscribe-Prozess
amqfcxba	Brokerworker-Prozess
amqrmppa	Prozess-Pooling-Prozess
amqcrsta	Nicht-Thread-Responder-Jobprozess
amqcrs6b	LU62 -Empfängerkanal und Clientverbindung
amqrrmfa	Der Repositoryprozess (für Cluster)
Amqzdmaa	Verzögerter Nachrichtenprozessor
amqpcsea	Befehlsserver
runmqtrm	Auslösermonitor für einen Server aufrufen
runmqdlq	Handler für dead-letter-Warteschlangen aufrufen
runmqchi	Der Kanalinitiatorprozess
runmqlsr	Der Kanal-Listener-Prozess

Anmerkung: Mit dem Befehl **`kill -9`** können Sie Prozesse beenden, die nicht gestoppt werden können.

Wenn Sie den Warteschlangenmanager manuell stoppen, werden möglicherweise FFSTs verwendet und die FDC-Dateien in `/var/mqm/errors.` betrachten dies nicht als Fehler im Warteschlangenmanager.

Der Warteschlangenmanager wird normal erneut gestartet, auch wenn er mit dieser Methode gestoppt wurde.

Lokale Verwaltungsaufgaben mithilfe von MQSC-Befehlen ausführen

Dieser Abschnitt enthält eine Einführung in MQSC-Befehle und beschreibt deren Verwendung für einige häufig vorkommende Tasks.

Wenn Sie IBM WebSphere MQ für Windows oder IBM WebSphere MQ für Linux (x86 -und x86-64 -Plattformen) verwenden, können Sie auch die in diesem Abschnitt beschriebenen Operationen mit IBM WebSphere MQ Explorer ausführen. Weitere Informationen finden Sie im Abschnitt „Verwaltung mit IBM WebSphere MQ Explorer" auf Seite 58.

Sie können MQSC-Befehle zum Verwalten von WS-Manager-Objekten verwenden, einschließlich des Warteschlangenmanagers selbst, Warteschlangen, Prozessdefinitionen, Kanäle, Clientverbindungskanäle, Empfangsprogramme, Services, Namenslisten, Cluster und Authentifizierungsinformationsobjekte. Dieser Abschnitt befasst sich mit Warteschlangenmanagern, Warteschlangen und Prozessdefinitionen. Informationen zur Verwaltung von Kanälen, Clientverbindungskanälen und Listenerobjekten finden Sie im Abschnitt Objekte . Informationen zu allen MQSC-Befehlen zum Verwalten von WS-Manager-Objekten finden Sie in „Scriptbefehle (MQSC)" auf Seite 76.

MQSC-Befehle werden mit dem Befehl `runmqsc` an einen Warteschlangenmanager ausgegeben. (Weitere Informationen zu diesem Befehl finden Sie in runmqsc .) Sie können dies interaktiv tun, Befehle über eine Tastatur ausgeben oder die Standardeingabeeinheit (`stdin`) umleiten, um eine Folge von Befehlen aus einer ASCII-Textdatei auszuführen. In beiden Fällen ist das Format der Befehle identisch. (Informationen

zum Ausführen der Befehle in einer Textdatei finden Sie im Abschnitt „MQSC-Befehle aus Textdateien ausführen” auf Seite 80.)

Sie können den Befehl `runmqsc` auf drei Arten ausführen, je nachdem, welche Flags im Befehl festgelegt sind:

- Überprüfen Sie einen Befehl, ohne ihn auszuführen. Dabei werden die MQSC-Befehle in einem lokalen WS-Manager geprüft, aber nicht ausgeführt.
- Führen Sie einen Befehl auf einem lokalen WS-Manager aus, auf dem die MQSC-Befehle auf einem lokalen Warteschlangenmanager ausgeführt werden.
- Führen Sie einen Befehl auf einem fernen WS-Manager aus, auf dem die MQSC-Befehle auf einem fernen Warteschlangenmanager ausgeführt werden.

Sie können den Befehl auch gefolgt von einem Fragezeichen ausführen, um die Syntax anzuzeigen.

Objektattribute, die in MQSC-Befehlen angegeben werden, werden in diesem Abschnitt in Großbuchstaben (z. B. RQMNAME) angezeigt, obwohl die Groß-/Kleinschreibung nicht beachtet werden muss. Die Namen von MQSC-Befehlsattributen sind auf acht Zeichen begrenzt. MQSC-Befehle sind auf anderen Plattformen verfügbar, einschließlich IBM i und z/OS.

MQSC-Befehle sind in der Themensammlung im Abschnitt MQSC-Referenz zusammengefasst.

Scriptbefehle (MQSC)

MQSC-Befehle stellen eine einheitliche Methode zur Ausgabe lesbarer Befehle auf WebSphere MQ -Plattformen bereit. Informationen zu Befehlen im *Programmable Command Format (PCF)* finden Sie im Abschnitt „Einführung in Programmierbare Befehlsformate” auf Seite 9.

Das allgemeine Format der Befehle wird im Abschnitt MQSC-Befehle beschrieben.

Bei Verwendung von MQSC-Befehlen sollten die folgenden Regeln beachtet werden:

- Jeder Befehl beginnt mit einem Primärparameter (ein Verb), und es folgt ein sekundärer Parameter (ein Substantiv). Darauf folgt dann der Name oder der generische Name des Objekts (in runden Klammern), wenn es einen Namen gibt, der in den meisten Befehlen vorhanden ist. Danach können die Parameter in der Regel in beliebiger Reihenfolge auftreten; wenn ein Parameter einen entsprechenden Wert hat, muss der Wert direkt nach dem Parameter, auf den er sich bezieht, auftreten.
- Schlüsselwörter, runde Klammern und Werte können durch eine beliebige Anzahl von Leerzeichen und Kommas voneinander getrennt werden. Ein Komma, das in den Syntaxdiagrammen angezeigt wird, kann immer durch ein oder mehrere Leerzeichen ersetzt werden. Jedem Parameter (nach dem primären Parameter) muss mindestens ein Leerzeichen unmittelbar vorangehen.
- Eine beliebige Anzahl von Leerzeichen kann am Anfang oder Ende des Befehls und zwischen Parametern, Interpunktionszeichen und Werten auftreten. Der folgende Befehl ist z. B. gültig:

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

  Leerzeichen in einem Paar von Anführungszeichen sind signifikant.
- Zusätzliche Kommas können überall dort angezeigt werden, wo Leerzeichen zulässig sind, und sie werden so behandelt, als wären sie leer (es sei denn, es handelt sich um Zeichenfolgen, die in Anführungszeichen eingeschlossen sind).
- Wiederholte Parameter sind nicht zulässig. Die Wiederholung eines Parameters mit seiner "NO" -Version, wie in REPLACE NOREPLACE, ist ebenfalls nicht zulässig.
- Zeichenfolgen, die Leerzeichen, Kleinbuchstaben oder andere Sonderzeichen enthalten als:
  - Punkt (.)
  - Schrägstrich (/)
  - Unterstrich (_)
  - Prozentzeichen (%)

müssen in einfache Anführungszeichen eingeschlossen werden, es sei denn, sie sind:

  - Generische Werte, die mit einem Stern enden
  - Ein einzelner Stern (z. B. TRACE(*))
  - Eine Bereichsangabe mit einem Doppelpunkt (z. B. CLASS(01:03))

  Wenn die Zeichenfolge selbst ein Hochkomma enthält, wird das Hochkomma durch zwei einfache Anführungszeichen dargestellt.Kleinbuchstaben, die nicht in Anführungszeichen enthalten sind, werden in Großbuchstaben gefaltet.
- Auf anderen Plattformen als z/OSwird eine Zeichenfolge, die keine Zeichen enthält (d. h. zwei einfache Anführungszeichen ohne Leerzeichen dazwischen), als Leerzeichen interpretiert, das in einfache Anführungszeichen eingeschlossen ist, d. h. auf dieselbe Weise wie ('') interpretiert wird. Eine Ausnahme ist, wenn das verwendete Attribut eines der folgenden Attribute ist:

  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR

  dann werden zwei einfache Anführungszeichen ohne Leerzeichen als Zeichenfolge mit Nulllänge interpretiert.
- In Version 7.0 behalten alle abschließenden Leerzeichen in Zeichenfolgeattributen, die auf MQCHARV-Typen basieren, also zum Beispiel bei Unterbenutzerdaten von SELECTOR, ihre Bedeutung, d. h. 'abc ' ist nicht das Gleiche wie 'abc'.
- Eine linke runde Klammer, gefolgt von einer rechten runden Klammer ohne signifikante Informationen dazwischen, z. B.

  ```
  NAME ( )
  ```

  ist nicht gültig, es sei denn, es ist ausdrücklich vermerkt.
- Bei Schlüsselwörtern muss die Groß-/Kleinschreibung nicht beachtet werden: AltER, alter und ALTER sind alle akzeptabel.Alles, was in Anführungszeichen nicht enthalten ist, wird in Großbuchstaben geschrieben.
- Synonyme werden für einige Parameter definiert.DEF ist z. B. immer ein Synonym für DEFINE, daher ist DEF QLOCAL gültig. Synonyme sind jedoch nicht nur Mindestzeichenfolgen; DEFI ist kein gültiges Synonym für DEFINE.

  Anmerkung: Es ist kein Synonym für den Parameter DELETE vorhanden. Dies soll verhindern, dass Objekte versehentlich gelöscht werden, wenn DEF, das Synonym für DEFINE, verwendet wird.

Eine Übersicht über die Verwendung von MQSC-Befehlen für die Verwaltung von IBM WebSphere MQ finden Sie in „Lokale Verwaltungsaufgaben mithilfe von MQSC-Befehlen ausführen” auf Seite 75.

MQSC-Befehle verwenden bestimmte Sonderzeichen, um bestimmte Bedeutungen zu haben. Weitere Informationen zu diesen Sonderzeichen und zur Verwendung dieser Sonderzeichen finden Sie im Abschnitt Zeichen mit besonderer Bedeutung .

Weitere Informationen zum Erstellen von Scripts mit Hilfe von MQSC-Befehlen finden Sie im Abschnitt Befehlsscripts erstellen .

Die vollständige Liste der MQSC-Befehle finden Sie im Abschnitt MQSC-Befehle .

Zugehörige Tasks
Befehlsscripts erstellen

WebSphere MQ -Objektnamen

Vorgehensweise beim Verwenden von Objektnamen in MQSC-Befehlen.

In den Beispielen verwenden wir einige lange Namen für Objekte. Dies soll Ihnen helfen, den Typ des Objekts zu identifizieren, mit dem Sie es zu tun haben.

Wenn Sie MQSC-Befehle absetzen, müssen Sie nur den lokalen Namen der Warteschlange angeben. In unseren Beispielen verwenden wir Warteschlangennamen, wie z. B.:

```
ORANGE.LOCAL.QUEUE
```

Der `LOCAL.QUEUE`-Teil des Namens soll veranschaulichen, dass es sich bei dieser Warteschlange um eine lokale Warteschlange handelt. Es ist ***nicht*** für die Namen von lokalen Warteschlangen im Allgemeinen erforderlich.

Der Name `saturn.queue.manager` wird auch als Name des Warteschlangenmanagers verwendet. Der `queue.manager`-Teil des Namens soll veranschaulichen, dass es sich bei diesem Objekt um einen Warteschlangenmanager handelt. Es ist *nicht* für die Namen von Warteschlangenmanagern im Allgemeinen erforderlich.

Groß-/Kleinschreibung in MQSC-Befehlen

MQSC-Befehle, einschließlich ihrer Attribute, können in Groß-oder Kleinbuchstaben geschrieben werden. Objektnamen in MQSC-Befehlen werden in Großbuchstaben geschrieben (also QUEUE und Warteschlange werden nicht unterschieden), es sei denn, die Namen sind in einfache Anführungszeichen eingeschlossen. Wenn Anführungszeichen nicht verwendet werden, wird das Objekt mit einem Namen in Großbuchstaben verarbeitet. Weitere Informationen finden Sie imWebSphere MQ-Scriptbefehlsreferenz (MQSC) .

Beim Aufruf des Befehls `runmqsc` muss in einigen WebSphere MQ -Umgebungen die Groß-/Kleinschreibung beachtet werden. Dies gilt für alle WebSphere MQ -Steuerbefehle. Weitere Informationen hierzu enthält der Abschnitt Steuerbefehle verwenden .

Standardeingabe und-ausgabe

Die *Standardeingabeeinheit* , die auch als `stdin` bezeichnet wird, ist die Einheit, von der die Eingabe für das System ausgeführt wird. In der Regel ist dies die Tastatur, aber Sie können angeben, dass die Eingabe beispielsweise aus einem seriellen Anschluss oder einer Plattendatei stammen soll. Die *Standardausgabeeinheit* , die auch als `stdout` bezeichnet wird, ist die Einheit, an die die Ausgabe des Systems gesendet wird. Normalerweise ist dies eine Anzeige, aber Sie können die Ausgabe an einen seriellen Anschluss oder an eine Datei umleiten.

Bei Betriebssystembefehlen und WebSphere MQ -Steuerbefehlen leitet der Operator < die Eingabe um. Wenn diesem Operator ein Dateiname gefolgt wird, wird die Eingabe aus der Datei übernommen. Entsprechend leitet der Operator > die Ausgabe um. Wenn diesem Operator ein Dateiname folgt, erfolgt die Ausgabe in die genannte Datei.

MQSC-Befehle interaktiv verwenden

Sie können MQSC-Befehle interaktiv verwenden, indem Sie ein Befehlsfenster oder eine -Shell verwenden.

Wenn Sie MQSC-Befehle interaktiv verwenden möchten, öffnen Sie ein Befehlsfenster oder eine Shell und geben Sie Folgendes ein:

```
runmqsc
```

In diesem Befehl wurde kein Warteschlangenmanagername angegeben, so dass die MQSC-Befehle vom Standardwarteschlangenmanager verarbeitet werden. Wenn Sie einen anderen WS-Manager verwenden möchten, geben Sie den Namen des WS-Managers im Befehl **runmqsc** an. Verwenden Sie beispielsweise den folgenden Befehl, um MQSC-Befehle auf WS-Manager `jupiter.queue.manager` auszuführen:

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

Danach werden alle MQSC-Befehle, die Sie eingeben, von diesem WS-Manager verarbeitet, vorausgesetzt, er befindet sich auf demselben Knoten und ist bereits aktiv.

Jetzt können Sie bei Bedarf alle MQSC-Befehle eingeben. Versuchen Sie es zum Beispiel wie folgt:

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

Verwenden Sie für Befehle, die zu viele Parameter haben, die in eine Zeile passen, Fortsetzungszeichen, um anzugeben, dass ein Befehl in der folgenden Zeile fortgesetzt wird:

- Ein Minuszeichen (-) gibt an, dass der Befehl ab dem Anfang der folgenden Zeile fortgesetzt werden soll.
- Ein Pluszeichen (+) gibt an, dass der Befehl aus dem ersten nicht leeren Zeichen in der folgenden Zeile fortgesetzt werden soll.

Die Befehlseingabe endet mit dem letzten Zeichen einer nicht leeren Zeile, die kein Fortsetzungszeichen ist. Sie können die Befehlseingabe auch explizit beenden, indem Sie ein Semikolon (;) eingeben. (Dies ist besonders nützlich, wenn Sie versehentlich ein Fortsetzungszeichen am Ende der letzten Zeile der Befehlseingabe eingeben.)

Feedback von MQSC-Befehlen

Wenn Sie MQSC-Befehle absetzen, gibt der WS-Manager Bedienernachrichten zurück, die Ihre Aktionen bestätigen oder Ihnen mitteilen, welche Fehler Sie gemacht haben. Beispiel:

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

Diese Nachricht bestätigt, dass eine Warteschlange erstellt wurde.

```
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
```

Diese Nachricht weist darauf hin, dass Sie einen Syntaxfehler gemacht haben.

Diese Nachrichten werden an die Standardausgabeeinheit gesendet. Wenn Sie den Befehl nicht korrekt eingegeben haben, lesen Sie die Informationen zur korrekten Syntax in WebSphere MQ-Scriptbefehlsreferenz (MQSC) .

Interaktive Eingabe von MQSC-Befehlen wird beendet

Geben Sie den Befehl END ein, um die Arbeit mit MQSC-Befehlen zu beenden.

Alternativ können Sie das EOF-Zeichen für Ihr Betriebssystem verwenden.

MQSC-Befehle aus Textdateien ausführen

Die Ausführung von MQSC-Befehlen ist interaktiv für Schnelltests geeignet. Wenn Sie jedoch über sehr lange Befehle verfügen oder eine bestimmte Folge von Befehlen wiederholt verwenden, sollten Sie die Umleitung von `stdin` aus einer Textdatei in Betracht ziehen.

„Standardeingabe und-ausgabe" auf Seite 78 enthält Informationen zu `stdin` und `stdout`. Um `stdin` aus einer Textdatei umzuleiten, müssen Sie zuerst mit Ihrem üblichen Texteditor eine Textdatei erstellen, die die MQSC-Befehle enthält. Wenn Sie den Befehl `runmqsc` verwenden, verwenden Sie die Umleitungsoperatoren. Mit dem folgenden Befehl wird beispielsweise eine Folge von Befehlen ausgeführt, die in der Textdatei `myprog.in` enthalten sind:

```
runmqsc < myprog.in
```

In ähnlicher Weise können Sie die Ausgabe auch in eine Datei umleiten. Eine Datei, die die MQSC-Befehle für die Eingabe enthält, wird als *MQSC-Befehlsdatei* bezeichnet. Die Ausgabedatei, die Antworten aus dem Warteschlangenmanager enthält, wird als *Ausgabedatei* bezeichnet.

Wenn Sie `stdin` und `stdout` im Befehl `runmqsc` umleiten möchten, verwenden Sie folgende Form des Befehls:

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

Dieser Befehl ruft die MQSC-Befehle in der MQSC-Befehlsdatei auf `myprog.in`. Da kein Warteschlangenmanagername angegeben wurde, werden die MQSC-Befehle für den Standardwarteschlangenmanager ausgeführt. Die Ausgabe wird an die Textdatei `myprog.out` gesendet.Abbildung 12 auf Seite 80 zeigt einen Auszug aus der MQSC-Befehlsdatei `myprog.in` und Abbildung 13 auf Seite 81 zeigt den entsprechenden Auszug der Ausgabe in `myprog.out`.

Um `stdin` und `stdout` im Befehl `runmqsc` für einen Warteschlangenmanager (`saturn.queue.manager`) umzuleiten, der nicht der Standardwert ist, verwenden Sie das folgende Format des Befehls:

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

MQSC-Befehlsdateien

MQSC-Befehle werden in lesbarer Form geschrieben, d. A. in ASCII-Text.Abbildung 12 auf Seite 80 ist ein Auszug aus einer MQSC-Befehlsdatei, der einen MQSC-Befehl (DEFINE QLOCAL) mit seinen Attributen zeigt. Die WebSphere MQ-Scriptbefehlsreferenz (MQSC) enthält eine Beschreibung jedes MQSC-Befehls und seine Syntax.

```
 .
 .
 .
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;
 .
 .
 .
```

Abbildung 12. Aus einer MQSC-Befehlsdatei extrahieren

Um die Portierbarkeit zwischen WebSphere MQ -Umgebungen zu gewährleisten, begrenzen Sie die Zeilenlänge in MQSC-Befehlsdateien auf 72 Zeichen. Das Pluszeichen gibt an, dass der Befehl in der nächsten Zeile fortgesetzt wird.

MQSC-Befehlsberichte

Der Befehl **runmqsc** gibt einen *Bericht* zurück, der an `stdout` gesendet wird. Der Bericht enthält:

- Ein Header, der MQSC-Befehle als Quelle für den Bericht identifiziert:

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

  Dabei ist `jupiter.queue.manager` der Name des Warteschlangenmanagers.
- Eine optionale nummerierte Liste der ausgegebenen MQSC-Befehle. Standardmäßig wird der Text der Eingabe an die Ausgabe zurückgemeldet. In dieser Ausgabe wird jeder Befehl durch eine Folgenummer als Präfix vorangestellt, wie in Abbildung 13 auf Seite 81 dargestellt. Sie können jedoch das Flag -e im Befehl `runmqsc` verwenden, um die Ausgabe zu unterdrücken.
- Es wurde eine Syntaxfehlernachricht für alle Befehle gefunden, die sich als fehlerhaft erwiesen haben.
- Eine *Bedienernachricht* , die das Ergebnis der Ausführung der einzelnen Befehle angibt. Die Bedienernachricht für den erfolgreichen Abschluss des Befehls DEFINE QLOCAL lautet z. B.:

  ```
  AMQ8006: WebSphere MQ queue created.
  ```
- Andere Nachrichten, die sich aus allgemeinen Fehlern beim Ausführen der Scriptdatei ergeben.
- Eine kurze statistische Zusammenfassung des Berichts, in der die Anzahl der gelesenen Befehle, die Anzahl der Befehle mit Syntaxfehlern und die Anzahl der Befehle angegeben sind, die nicht verarbeitet werden konnten.

  Anmerkung: Der WS-Manager versucht, nur die Befehle zu verarbeiten, die keine Syntaxfehler aufweisen.

```
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.
      :
 .
 .
```

Abbildung 13. Auszug aus einer MQSC-Befehlsberichtsdatei

Die bereitgestellten MQSC-Befehlsdateien ausführen

Die folgenden MQSC-Befehlsdateien werden mit WebSphere MQbereitgestellt:

amqscos0.tst
Definitionen von Objekten, die von Beispielprogrammen verwendet werden.

amqscic0.tst
Definitionen von Warteschlangen für CICS -Transaktionen.

In WebSphere MQ für Windowsbefinden sich diese Dateien im Verzeichnis *MQ_INSTALLATION_PATH*\tools\mqsc\samples. *MQ_INSTALLATION_PATH* steht für das übergeordnete Verzeichnis, in dem WebSphere MQ installiert ist.

Auf UNIX and Linux -Systemen befinden sich diese Dateien im Verzeichnis *MQ_INSTALLATION_PATH*/samp. *MQ_INSTALLATION_PATH* steht für das übergeordnete Verzeichnis, in dem WebSphere MQ installiert ist.

Der Befehl, der sie ausführt, lautet:

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

Befehle 'runmqsc' zum Prüfen von Befehlen verwenden

Mit dem Befehl runmqsc können Sie MQSC-Befehle auf einem lokalen Warteschlangenmanager überprüfen, ohne sie tatsächlich auszuführen. Setzen Sie dazu das Flag "-v" im Befehl runmqsc . Beispiel:

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

Wenn Sie runmqsc für eine MQSC-Befehlsdatei aufrufen, überprüft der Warteschlangenmanager jeden Befehl und gibt einen Bericht zurück, ohne die MQSC-Befehle tatsächlich auszuführen. Auf diese Weise können Sie die Syntax der Befehle in Ihrer Befehlsdatei überprüfen. Dies ist besonders wichtig, wenn Sie die folgenden Schritte ausführen:

- Es wird eine große Anzahl Befehle aus einer Befehlsdatei ausgeführt.
- Die Verwendung einer MQSC-Befehlsdatei ist mehrfach zu überdauern.

Der zurückgegebene Bericht ähnelt dem in Abbildung 13 auf Seite 81 dargestellten Bericht.

Sie können diese Methode nicht verwenden, um MQSC-Befehle über Remotezugriff zu überprüfen. Beispiel: Wenn Sie versuchen, diesen Befehl auszuführen:

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

Das Flag -w, das Sie verwenden, um anzugeben, dass der Warteschlangenmanager fern ist, wird ignoriert, und der Befehl wird lokal im Prüfmodus ausgeführt. 30 ist die Anzahl der Sekunden, die WebSphere MQ auf Antworten vom fernen Warteschlangenmanager wartet.

MQSC-Befehle aus Stapeldateien ausführen

Wenn Sie über sehr lange Befehle verfügen oder eine bestimmte Folge von Befehlen wiederholt verwenden, sollten Sie die Umleitung von stdin aus einer Stapeldatei in Erwägung ziehen.

Um stdin aus einer Stapeldatei umzuleiten, müssen Sie zuerst eine Stapeldatei erstellen, die die MQSC-Befehle enthält, und verwenden Sie dabei Ihren üblichen Texteditor. Wenn Sie den Befehl runmqsc verwenden, verwenden Sie die Umleitungsoperatoren. Das folgende Beispiel:

1. Erstellt einen Testwarteschlangenmanager TESTQM
2. Erstellt eine übereinstimmende CLNTCONN-und Listener-Gruppe für die Verwendung von TCP/IP-Port 1600
3. Erstellt eine Testwarteschlange, TESTQ
4. Versetzt eine Nachricht in die Warteschlange mit Hilfe des Beispielprogramms '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
```

Abbildung 14. Beispielscript für die Ausführung von MQSC-Befehlen aus einer Stapeldatei

Probleme mit MQSC-Befehlen beheben

Wenn die Ausführung von MQSC-Befehlen nicht möglich ist, verwenden Sie die Informationen in diesem Thema, um festzustellen, ob eine dieser allgemeinen Probleme für Sie zutreffen. Es ist nicht immer klar, was das Problem ist, wenn Sie den Fehler lesen, den ein Befehl generiert.

Wenn MQSC-Befehle nicht ausgeführt werden können, verwenden Sie die folgenden Informationen, um festzustellen, ob eines dieser häufig auftretenden Probleme auf Sie zutrifft. Es ist nicht immer offensichtlich, welches Problem beim Lesen des generierten Fehlers auftritt.

Beachten Sie bei der Verwendung des Befehls `runmqsc` Folgendes:

- Mit dem Operator **<** können Sie die Eingabe aus einer Datei umleiten. Wenn Sie diesen Operator weglassen, interpretiert der WS-Manager den Dateinamen als WS-Managernamen und gibt die folgende Fehlernachricht aus:

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

- Wenn Sie die Ausgabe in eine Datei umleiten, verwenden Sie den Umleitungsoperator **>**. Standardmäßig wird die Datei zu dem Zeitpunkt im aktuellen Arbeitsverzeichnis gespeichert, an dem `runmqsc` aufgerufen wird. Geben Sie einen vollständig qualifizierten Dateinamen an, um die Ausgabe an eine bestimmte Datei und ein bestimmtes Verzeichnis zu senden.
- Überprüfen Sie, ob Sie den Warteschlangenmanager erstellt haben, der die Befehle ausführen soll, indem Sie den folgenden Befehl verwenden, um alle Warteschlangenmanager anzuzeigen:

  ```
  dspmq
  ```

- Der WS-Manager muss aktiv sein. Wenn dies nicht der Fall ist, starten Sie es; (siehe Warteschlangenmanager starten). Sie erhalten eine Fehlernachricht, wenn Sie versuchen, einen Warteschlangenmanager zu starten, der bereits ausgeführt wird.
- Geben Sie im Befehl `runmqsc` einen Warteschlangenmanagernamen an, wenn Sie keinen Standardwarteschlangenmanager definiert haben oder der folgende Fehler auftritt:

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

- Sie können keinen MQSC-Befehl als Parameter des Befehls `runmqsc` angeben. Dies ist z. B. nicht zulässig:

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

- Sie können keine MQSC-Befehle eingeben, bevor Sie den Befehl `runmqsc` ausgeben.
- Sie können keine Steuerbefehle über `runmqsc`ausführen. Sie können beispielsweise nicht den Befehl `strmqm` ausgeben, um einen Warteschlangenmanager zu starten, während Sie MQSC-Befehle interaktiv ausführen. Wenn Sie dies tun, erhalten Sie Fehlernachrichten ähnlich der folgenden:

```
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
```

Mit Warteschlangenmanagern arbeiten

Beispiele für MQSC-Befehle, die Sie zum Anzeigen oder Ändern von Warteschlangenmanagerattributen verwenden können.

WS-Manager-Attribute anzeigen

Mit dem folgenden MQSC-Befehl können Sie die Attribute des im Befehl **runmqsc** angegebenen Warteschlangenmanagers anzeigen:

```
DISPLAY QMGR
```

Die typische Ausgabe dieses Befehls wird in Abbildung 15 auf Seite 85 angezeigt.

```
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)
```

Abbildung 15. Typische Ausgabe des Befehls DISPLAY QMGR

Der Parameter ALL ist der Standardwert für den Befehl DISPLAY QMGR . Es zeigt alle Attribute des Warteschlangenmanagers an. In der Ausgabe werden insbesondere der Name des Standardwarteschlangenmanagers, der Name der Warteschlange für dead-Mail und der Name der Befehlswarteschlange angegeben.

Sie können bestätigen, dass diese Warteschlangen vorhanden sind, indem Sie den folgenden Befehl eingeben:

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

Daraufhin wird eine Liste der Warteschlangen angezeigt, die dem Stamm SYSTEM.*entsprechen. Die runden Klammern sind erforderlich.

Ändern von WS-Managerattributen

Um die Attribute des im Befehl **runmqsc** angegebenen Warteschlangenmanagers zu ändern, geben Sie mit dem MQSC-Befehl ALTER QMGRdie Attribute und Werte an, die Sie ändern möchten. Verwenden Sie z. B. die folgenden Befehle, um die Attribute von jupiter.queue.manager zu ändern:

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

Der Befehl ALTER QMGR ändert die verwendete Warteschlange für nicht zustellbare Nachrichten und aktiviert Sperrereignisse.

Zugehörige Verweise

Attribute für den Warteschlangenmanager

Mit lokalen Warteschlangen arbeiten

Dieser Abschnitt enthält Beispiele für einige MQSC-Befehle, die Sie zum Verwalten von lokalen, Modell- und Aliaswarteschlangen verwenden können.

Ausführliche Informationen zu diesen Befehlen finden Sie im WebSphere MQ-Scriptbefehlsreferenz (MQSC) .

Lokale Warteschlange definieren

Bei einer Anwendung ist der lokale WS-Manager der Warteschlangenmanager, mit dem die Anwendung verbunden ist. Warteschlangen, die vom lokalen WS-Manager verwaltet werden, werden als lokale Warteschlangen für diesen Warteschlangenmanager angegeben.

Mit dem MQSC-Befehl DEFINE QLOCAL können Sie eine lokale Warteschlange erstellen. Sie können auch den Standardwert verwenden, der in der lokalen Standardwarteschlangendefinition definiert ist, oder Sie können die Warteschlangenkenndaten aus denen der lokalen Standardwarteschlange ändern.

Anmerkung: Die lokale Standardwarteschlange hat den Namen SYSTEM.DEFAULT.LOCAL.QUEUE, und sie wurde bei der Systeminstallation erstellt.

Der folgende Befehl DEFINE QLOCAL definiert beispielsweise eine Warteschlange mit dem Namen ORANGE.LOCAL.QUEUE mit folgenden Merkmalen:

- Sie ist aktiviert für Abrufe, aktiviert für die Option "puts" und arbeitet mit einer Prioritätsauftragsbasis.
- Es handelt sich um eine *normale* Warteschlange; es handelt sich nicht um eine Initialisierungswarteschlange oder eine Übertragungswarteschlange, und es werden keine Auslösenachrichten generiert.
- Die maximale Warteschlangenlänge beträgt 5000 Nachrichten; die maximale Nachrichtenlänge beträgt 4194304 Byte.

```
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);
```

Anmerkung:

1. Mit Ausnahme des Werts für die Beschreibung sind alle gezeigten Attributwerte die Standardwerte. Sie wurden hier nur zur Veranschaulichung dargestellt. Sie können sie übergehen, wenn Sie sicher sind, dass die Standardwerte die gewünschten Werte sind oder nicht geändert wurden. Weitere Informationen hierzu finden Sie im Abschnitt „Standardobjektattribute anzeigen” auf Seite 87.
2. USAGE (NORMAL) gibt an, dass diese Warteschlange keine Übertragungswarteschlange ist.
3. Wenn bereits eine lokale Warteschlange auf demselben WS-Manager mit dem Namen ORANGE.LOCAL.QUEUE vorhanden ist, schlägt dieser Befehl fehl. Verwenden Sie das Attribut REPLACE, wenn Sie die vorhandene Definition einer Warteschlange überschreiben möchten. Lesen Sie dazu auch „Attribute der lokalen Warteschlange ändern” auf Seite 88.

Warteschlange für dead-Mail definieren

Jeder WS-Manager muss über eine lokale Warteschlange verfügen, die als Warteschlange für einen dead-letter verwendet werden muss, damit Nachrichten, die nicht an ihr korrektes Ziel zugestellt werden können, für einen späteren Abruf gespeichert werden können. Sie müssen dem Warteschlangenmanager die Warteschlange für nicht zustellbare Nachrichten mitteilen.

Um den Warteschlangenmanager über die Warteschlange für nicht zustellbare Nachrichten zu informieren, geben Sie im Befehl `crtmqm` (z. B.`crtmqm -u DEAD.LETTER.QUEUE`) einen Namen für die Warteschlange für nicht zustellbare Nachrichten an oder verwenden Sie das Attribut DEADQ im Befehl ALTER QMGR, um später einen Namen anzugeben. Sie müssen die Warteschlange für den dead-letter definieren, bevor Sie sie verwenden.

Es ist eine Beispiel-Deadletter-Warteschlange mit dem Namen SYSTEM.DEAD.LETTER.QUEUE mit dem Produkt verfügbar. Diese Warteschlange wird bei der Erstellung des Warteschlangenmanagers automatisch erstellt. Sie können diese Definition bei Bedarf ändern und sie umbenennen.

Eine Warteschlange für nicht zustellbare Nachrichten hat keine besonderen Anforderungen mit Ausnahme der folgenden:

- Es muss sich um eine lokale Warteschlange handeln.
- Das Attribut MAXMSGL (maximale Nachrichtenlänge) muss die Warteschlange aktivieren, damit die größten Nachrichten, die der Warteschlangenmanager verarbeiten muss, **plus** die Größe des Headers "dead-letter" (MQDLH) erfüllen müssen.

WebSphere MQ stellt eine Steuerroutine der Warteschlange für nicht zustellbare Nachrichten bereit, mit der Sie angeben können, wie in einer Warteschlange für nicht zustellbare Nachrichten gefundene Nachrichten verarbeitet oder entfernt werden. Weitere Informationen finden Sie unter Behandlung nicht zugestellter Nachrichten mit der WebSphere MQ -Steuerroutine der Warteschlange für nicht zustellbare Nachrichten .

Standardobjektattribute anzeigen

Mit dem Befehl DISPLAY QUEUE können Sie Attribute anzeigen, die aus dem Standardobjekt übernommen wurden, als ein WebSphere MQ -Objekt definiert wurde.

Wenn Sie ein WebSphere MQ -Objekt definieren, werden alle Attribute, die Sie nicht angeben, aus dem Standardobjekt übernommen. Wenn Sie beispielsweise eine lokale Warteschlange definieren, übernimmt die Warteschlange alle Attribute, die Sie in der Definition aus der lokalen Standardwarteschlange weglassen, die als SYSTEM.DEFAULT.LOCAL.QUEUE bezeichnet wird. Um genau zu sehen, welche Attribute diese Attribute haben, verwenden Sie den folgenden Befehl:

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

Die Syntax dieses Befehls unterscheidet sich von der des entsprechenden DEFINE-Befehls. Im Befehl DISPLAY können Sie nur den Warteschlangennamen angeben, während Sie im Befehl DEFINE auch den Warteschlangentyp, d. h. QLOCAL, QALIAS, QMODEL oder QREMOTE, angeben müssen.

Sie können Attribute selektiv anzeigen, indem Sie sie einzeln angeben. Beispiel:

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

Mit diesem Befehl werden die drei angegebenen Attribute wie folgt angezeigt:

```
AMQ8409: Display Queue details.
   QUEUE(ORANGE.LOCAL.QUEUE)               TYPE(QLOCAL)
   CURDEPTH(0)                             MAXDEPTH(5000)
   MAXMSGL(4194304)
```

CURDEPTH ist die aktuelle Warteschlangenlänge, d. h. die Anzahl der Nachrichten in der Warteschlange. Dies ist ein nützliches Attribut, das angezeigt werden kann, da durch die Überwachung der Warteschlangenlänge sichergestellt werden kann, dass die Warteschlange nicht voll wird.

Lokale Warteschlangendefinition kopieren

Sie können eine Warteschlangendefinition mit dem Attribut LIKE im Befehl DEFINE kopieren.

Beispiel:

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

Mit diesem Befehl wird eine Warteschlange mit denselben Attributen wie die ursprüngliche Warteschlange ORANGE.LOCAL.QUEUE erstellt, und nicht die Attribute der lokalen Standardwarteschlange. Geben Sie den Namen der Warteschlange, die kopiert werden soll, ***genau*** so ein, wie er beim Erstellen der Warteschlange eingegeben wurde. Wenn der Name Kleinbuchstaben enthält, schließen Sie den Namen in einfache Anführungszeichen ein.

Sie können diese Form des Befehls DEFINE auch verwenden, um eine Warteschlangendefinition zu kopieren und dabei einzelne Attribute des Originals ersetzen. Beispiel:

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

Dieser Befehl kopiert die Attribute der Warteschlange ORANGE.LOCAL.QUEUE in die Warteschlange THIRD.QUEUE, gibt jedoch an, dass die maximale Nachrichtenlänge in der neuen Warteschlange 1024 Byte und nicht 4194304 betragen soll.

Anmerkung:

1. Wenn Sie das Attribut LIKE in einem DEFINE-Befehl verwenden, kopieren Sie die Warteschlangenattribute nur. Die Nachrichten werden nicht in die Warteschlange kopiert.
2. Wenn Sie eine lokale Warteschlange definieren, ohne LIKE anzugeben, ist sie mit der Angabe von DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE) identisch.

Attribute der lokalen Warteschlange ändern

Sie können die Warteschlangenattribute auf zwei Arten ändern, indem Sie entweder den Befehl ALTER QLOCAL oder den Befehl DEFINE QLOCAL mit dem Attribut REPLACE verwenden.

In „Lokale Warteschlange definieren” auf Seite 86 wurde die Warteschlange mit dem Namen ORANGE.LOCAL.QUEUE definiert. Nehmen wir zum Beispiel an, dass Sie die maximale Nachrichtenlänge in dieser Warteschlange auf 10.000 Byte reduzieren möchten.

- Befehl ALTER verwenden:

  ```
  ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
  ```

  Mit diesem Befehl wird ein einzelnes Attribut geändert, das der maximalen Nachrichtenlänge entspricht. Alle anderen Attribute bleiben unverändert.
- Befehl DEFINE mit der Option REPLACE verwenden, z. B.:

  ```
  DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
  ```

  Mit diesem Befehl werden nicht nur die maximale Nachrichtenlänge, sondern auch alle anderen Attribute geändert, die ihre Standardwerte erhalten. Die Warteschlange ist jetzt aktiviert, während sie zuvor gesperrt wurde. Put aktiviert ist der Standardwert, wie er in der Warteschlange SYSTEM.DEFAULT.LOCAL.QUEUE angegeben ist.

  Wenn Sie die maximale Nachrichtenlänge in einer vorhandenen Warteschlange ***verringern*** , sind die vorhandenen Nachrichten nicht betroffen. Alle neuen Nachrichten müssen jedoch die neuen Kriterien erfüllen.

Löschen einer lokalen Warteschlange

Sie können den Befehl CLEAR verwenden, um eine lokale Warteschlange zu löschen.

Wenn Sie alle Nachrichten aus einer lokalen Warteschlange mit dem Namen MAGENTA.QUEUE löschen möchten, verwenden Sie den folgenden Befehl:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

Anmerkung: Es gibt keine Eingabeaufforderung, mit der Sie Ihre Meinung ändern können. Sobald Sie die Eingabetaste drücken, gehen die Nachrichten verloren.

Sie können eine Warteschlange nicht löschen, wenn:

- Es sind nicht festgeschriebene Nachrichten vorhanden, die unter Synchronisationspunkt in die Warteschlange gestellt wurden.
- Eine Anwendung hat momentan die Warteschlange geöffnet.

Lokale Warteschlange löschen

Sie können den MQSC-Befehl DELETE QLOCAL verwenden, um eine lokale Warteschlange zu löschen.

Eine Warteschlange kann nicht gelöscht werden, wenn sie nicht festgeschriebene Nachrichten enthält. Wenn die Warteschlange aber mindestens eine festgeschriebene Nachricht und keine nicht festgeschriebenen Nachrichten enthält, kann sie nur gelöscht werden, wenn im Befehl die Option PURGE angegeben wird. Beispiel:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Indem Sie NOPURGE statt PURGE angeben, können Sie sicherstellen, dass die Warteschlange nicht gelöscht wird, wenn sie festgeschriebene Nachrichten enthält.

Warteschlangen durchsuchen

WebSphere MQ stellt einen Beispielwarteschlangenbrowser bereit, mit dem Sie den Inhalt der Nachrichten in einer Warteschlange anzeigen können. Der Browser wird sowohl in der Quellen-als auch in der ausführbaren Datei bereitgestellt.

MQ_INSTALLATION_PATH steht für das übergeordnete Verzeichnis, in dem WebSphere MQ installiert ist.

In WebSphere MQ für Windowslauten die Dateinamen und Pfade für den Beispielwarteschlangenbrowser wie folgt:

Quelle
MQ_INSTALLATION_PATH\tools\c\samples\

Ausführbare Datei
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

In WebSphere MQ für UNIX and Linuxlauten die Dateinamen und Pfade wie folgt:

Quelle
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

Ausführbare Datei
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

Für das Beispiel sind zwei Eingabeparameter erforderlich: der Warteschlangenname und der Name des Warteschlangenmanagers. Beispiel:

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

Typische Ergebnisse dieses Befehls werden in Abbildung 16 auf Seite 90 angezeigt.

```
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
```

Abbildung 16. Typische Ergebnisse aus dem Warteschlangenbrowser

Große Warteschlangen aktivieren

IBM WebSphere MQ unterstützt Warteschlangen, die größer als 2 GB sind.

Auf Windows-Systemen steht die Unterstützung für große Dateien ohne zusätzliche Unterstützung zur Verfügung. Auf AIX-, HP-UX-, Linux-und Solaris -Systemen müssen Sie die Unterstützung für große Dateien explizit aktivieren, damit Sie Warteschlangendateien erstellen können, die größer als 2 GB sind. Weitere Informationen hierzu finden Sie in der Dokumentation zum Betriebssystem.

Einige Dienstprogramme, wie z. B. 'tar', können Dateien mit mehr als 2 GB nicht bewältigen. Überprüfen Sie vor der Aktivierung der Unterstützung für große Dateien die Dokumentation zum Betriebssystem auf Informationen zu Einschränkungen bei den Dienstprogrammen, die Sie verwenden.

Informationen zur Planung der Speichermenge, die Sie für Warteschlangen benötigen, finden Sie auf der IBM WebSphere MQ Website für plattformspezifische Leistungsberichte:

```
https://www.ibm.com/software/integration/ts/mqseries/
```

Mit Aliaswarteschlangen arbeiten

Sie können eine Aliaswarteschlange definieren, um indirekt auf eine andere Warteschlange oder ein anderes Thema zu verweisen.

V 7.5.0.8

Achtung: Verteilerlisten unterstützen nicht die Verwendung von Aliaswarteschlangen, die auf Themenobjekte verweisen. Ab Version 7.5.0, Fix Pack 8 gibt IBM WebSphere MQ in dem Fall, dass eine Aliaswarteschlange auf ein Topic-Objekt in einer Verteilerliste verweist, den Fehler MQRC_ALIAS_BASE_Q_TYPE_ERROR zurück.

Für die Warteschlange, auf die sich eine Aliaswarteschlange bezieht, kann es sich um eine der folgenden Warteschlangen handeln:

- Eine lokale Warteschlange (siehe „Lokale Warteschlange definieren” auf Seite 86).
- Eine lokale Definition einer fernen Warteschlange (siehe „Lokale Definition einer fernen Warteschlange erstellen” auf Seite 116).
- Ein Thema.

Eine Aliaswarteschlange ist keine echte Warteschlange, sondern eine Definition, die zur Ausführungszeit in eine reale Warteschlange (oder Zielwarteschlange) aufgelöst wird. Die Definition der Aliaswarteschlange gibt die Zielwarteschlange an. Wenn eine Anwendung einen MQOPEN -Aufruf an eine Aliaswarteschlange ausgibt, löst der WS-Manager den Aliasnamen in den Zielwarteschlangennamen auf.

Eine Aliaswarteschlange kann nicht in eine andere lokal definierte Aliaswarteschlange aufgelöst werden. Eine Aliaswarteschlange kann jedoch in Aliaswarteschlangen aufgelöst werden, die an anderer Stelle in Clustern definiert sind, in denen der lokale WS-Manager Mitglied ist. Weitere Informationen finden Sie in der Namensauflösung .

Aliaswarteschlangen sind nützlich für:

- verschiedenen Anwendungen unterschiedliche Ebenen von Zugriffsberechtigungen für die Zielwarteschlange geben.
- Es können verschiedene Anwendungen auf unterschiedliche Weise mit derselben Warteschlange arbeiten. (Möglicherweise möchten Sie andere Standardprioritäten oder unterschiedliche Standardpersistenzwerte zuordnen.)
- Vereinfachung der Wartung, Migration und Lastverteilung. (Möglicherweise möchten Sie den Namen der Zielwarteschlange ändern, ohne Ihre Anwendung ändern zu müssen, die weiterhin den Aliasnamen verwendet.)

Angenommen, es wurde eine Anwendung entwickelt, um Nachrichten in eine Warteschlange mit dem Namen MY.ALIAS.QUEUE zu stellen. Sie gibt den Namen der Warteschlange an, wenn sie eine MQOPEN -Anforderung stellt, und indirekt, wenn sie eine Nachricht in diese Warteschlange einreiht. Der Anwendung ist nicht bekannt, dass es sich bei der Warteschlange um eine Aliaswarteschlange handelt. Für jeden MQI-Aufruf, der diesen Aliasnamen verwendet, löst der Warteschlangenmanager den Namen der realen Warteschlange auf, die entweder eine lokale Warteschlange oder eine ferne Warteschlange sein kann, die in diesem WS-Manager definiert ist.

Wenn Sie den Wert des Attributs TARGET ändern, können Sie MQI-Aufrufe an eine andere Warteschlange umleiten, möglicherweise auf einem anderen Warteschlangenmanager. Dies ist hilfreich bei der Wartung, Migration und Lastverteilung.

Definieren einer Aliaswarteschlange

Mit dem folgenden Befehl wird eine Aliaswarteschlange erstellt:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

Mit diesem Befehl werden MQI-Aufrufe umgeleitet, die MY.ALIAS.QUEUE in der Warteschlange YELLOW.QUEUE angeben. Der Befehl erstellt die Zielwarteschlange nicht; die MQI-Aufrufe schlagen fehl, wenn die Warteschlange YELLOW.QUEUE zur Ausführungszeit nicht vorhanden ist.

Wenn Sie die Aliasdefinition ändern, können Sie die MQI-Aufrufe an eine andere Warteschlange umleiten. For example:

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

Mit diesem Befehl werden MQI-Aufrufe an eine andere Warteschlange, MAGENTA.QUEUE, umgeleitet.

Sie können auch Aliaswarteschlangen verwenden, um eine einzelne Warteschlange (die Zielwarteschlange) zu erstellen, um unterschiedliche Attribute für verschiedene Anwendungen zu haben. Definieren Sie dies, indem Sie zwei Aliasnamen definieren, eine für jede Anwendung. Angenommen, es gibt zwei Anwendungen:

- Die Anwendung ALPHA kann Nachrichten in YELLOW.QUEUE einlegen, aber es ist nicht zulässig, Nachrichten von ihr abzurufen.
- Die Anwendung BETA kann Nachrichten von YELLOW.QUEUE abrufen, aber es ist nicht zulässig, Nachrichten in diese Datei zu stellen.

Mit dem folgenden Befehl wird ein Aliasname definiert, der aktiviert ist und für die Anwendung ALPHA inaktiviert wird:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (ENABLED) +
       GET (DISABLED)
```

Mit dem folgenden Befehl wird ein Alias definiert, der inaktiviert ist und für die Anwendung BETA aktiviert wird:

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (DISABLED) +
       GET (ENABLED)
```

ALPHA verwendet in seinen MQI-Aufrufen den Warteschlangennamen ALPHAS.ALIAS.QUEUE; BETA verwendet den Warteschlangennamen BETAS.ALIAS.QUEUE. Beide greifen auf die gleiche Warteschlange zu, aber auf unterschiedliche Weise.

Sie können die Attribute LIKE und REPLACE verwenden, wenn Sie Warteschlangenaliasnamen auf die gleiche Weise definieren, wie Sie diese Attribute mit lokalen Warteschlangen verwenden.

Andere Befehle mit Aliaswarteschlangen verwenden

Sie können die entsprechenden MQSC-Befehle verwenden, um Aliaswarteschlangenattribute anzuzeigen oder zu ändern oder um das Aliaswarteschlangenobjekt zu löschen. For example:

Verwenden Sie den folgenden Befehl, um die Attribute der Aliaswarteschlange anzuzeigen:

```
DISPLAY QALIAS (ALPHAS.ALIAS.QUEUE)
```

Verwenden Sie den folgenden Befehl, um den Namen der Basiswarteschlange zu ändern, in den der Aliasname aufgelöst wird, wobei die Änderung mit der Option `force` auch dann erzwungen wird, wenn die Warteschlange geöffnet ist:

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGET(ORANGE.LOCAL.QUEUE) FORCE
```

Verwenden Sie den folgenden Befehl, um diese Aliaswarteschlange zu löschen:

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

Sie können eine Aliaswarteschlange nicht löschen, wenn eine Anwendung momentan die Warteschlange geöffnet hat. Weitere Informationen zu diesem und anderen Befehlen für Aliaswarteschlangen finden Sie im WebSphere MQ-Scriptbefehlsreferenz (MQSC) .

Mit Modellwarteschlangen arbeiten

Ein Warteschlangenmanager erstellt eine *dynamische Warteschlange* , wenn er einen MQI-Aufruf von einer Anwendung empfängt, die einen Warteschlangennamen angibt, der als Modellwarteschlange definiert wurde. Der Name der neuen dynamischen Warteschlange wird vom WS-Manager beim Erstellen der Warteschlange generiert. Eine *Modellwarteschlange* ist eine Schablone, die die Attribute aller dynamischen Warteschlangen angibt, die aus dieser Warteschlange erstellt wurden. Modellwarteschlangen stellen eine komfortable Methode für Anwendungen zur Verfügung, um Warteschlangen nach Bedarf zu erstellen.

Modellwarteschlange definieren

Sie definieren eine Modellwarteschlange mit einer Gruppe von Attributen in derselben Weise, wie Sie eine lokale Warteschlange definieren. Modellwarteschlangen und lokale Warteschlangen verfügen über dieselbe Gruppe von Attributen, mit der Ausnahme, dass Sie in Modellwarteschlangen angeben können, ob die erstellten dynamischen Warteschlangen temporär oder permanent sind. (Permanente Warteschlangen werden über WS-Manager-Neustarts gepflegt, temporäre werden nicht ausgeführt.) Beispiel:

```
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)
```

Mit diesem Befehl wird eine Modellwarteschlangendefinition erstellt. Mit dem Attribut DEFTYPE wird festgelegt, dass die auf Basis dieser Vorlage erstellten Warteschlangen als permanente dynamische Warteschlangen erstellt werden. Alle Attribute, die nicht angegeben sind, werden automatisch aus der Standardwarteschlange SYSYTEM.DEFAULT.MODEL.QUEUE kopiert.

Sie können die Attribute LIKE und REPLACE beim Definieren von Modellwarteschlangen auf die gleiche Weise wie bei lokalen Warteschlangen verwenden.

Andere Befehle mit Modellwarteschlangen verwenden

Sie können die entsprechenden MQSC-Befehle verwenden, um die Attribute einer Modellwarteschlange anzuzeigen oder zu ändern oder um das Modellwarteschlangenobjekt zu löschen. Beispiel:

Mit dem folgenden Befehl können Sie die Attribute der Modellwarteschlange anzeigen:

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

Verwenden Sie den folgenden Befehl, um das Modell so zu ändern, dass es in jede dynamische Warteschlange, die aus diesem Modell erstellt wird, versetzt wird:

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

Mit dem folgenden Befehl können Sie diese Modellwarteschlange löschen:

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

Mit Verwaltungsthemen arbeiten

Verwenden Sie MQSC-Befehle, um Verwaltungsthemen zu verwalten.

Ausführliche Informationen zu diesen Befehlen finden Sie unter WebSphere MQ-Scriptbefehlsreferenz (MQSC) .

Zugehörige Konzepte

Verwaltungsthemenobjekte

„Verwaltungsthema definieren” auf Seite 94
Mit dem MQSC-Befehl **DEFINE TOPIC** können Sie ein Verwaltungsthema erstellen. Wenn Sie ein Verwaltungsthema definieren, können Sie optional jedes Themenattribut festlegen.

„Attribute des Verwaltungsthemenobjekts anzeigen” auf Seite 95
Mit dem MQSC-Befehl **DISPLAY TOPIC** können Sie ein Verwaltungsthemenobjekt anzeigen.

„Verwaltungsthemenattribute ändern” auf Seite 95
Sie können Themenattribute auf zwei Arten ändern: mit dem Befehl **ALTER TOPIC** oder mit dem Befehl **DEFINE TOPIC** und dem Attribut **REPLACE** .

„Eine Verwaltungsthemendefinition kopieren” auf Seite 96
Sie können eine Themendefinition mit dem Attribut LIKE im Befehl **DEFINE** kopieren.

„Löschen einer Verwaltungsthemendefinition” auf Seite 96
Mit dem MQSC-Befehl **DELETE TOPIC** können Sie ein Verwaltungsthema löschen.

Verwaltungsthema definieren

Mit dem MQSC-Befehl **DEFINE TOPIC** können Sie ein Verwaltungsthema erstellen. Wenn Sie ein Verwaltungsthema definieren, können Sie optional jedes Themenattribut festlegen.

Jedes Attribut des Themas, das nicht explizit festgelegt wurde, wird aus dem Standardverwaltungsthema SYSTEM.DEFAULT.TOPIC übernommen, das bei der Installation der Systeminstallation erstellt wurde.

Der folgende Befehl **DEFINE TOPIC** definiert beispielsweise ein Thema namens **ORANGE.TOPIC** mit den folgenden Merkmalen:

- Löst den Wert für die Themenzeichenfolge ORANGE auf. Informationen dazu, wie Themenzeichenfolgen verwendet werden können, finden Sie unter Themenzeichenfolgen kombinieren.
- Jedes Attribut, das auf "ASPARENT" gesetzt ist, verwendet das Attribut, wie es durch das übergeordnete Thema dieses Themas definiert ist. Diese Aktion wird die Themenstruktur so weit wiederholt, bis das Stammthema SYSTEM.BASE.TOPIC gefunden wird. Weitere Informationen zu Themenstrukturen finden Sie unter Themenstrukturen.

```
DEFINE TOPIC (ORANGE.TOPIC) +
       TOPICSTR (ORANGE) +
       DEFPRTY(ASPARENT) +
       NPMSGDLV(ASPARENT)
```

Anmerkung:

- Mit Ausnahme des Werts für die Themenzeichenfolge sind alle angezeigten Attributwerte die Standardwerte. Sie sind hier nur als Illustration dargestellt.Sie können sie übergehen, wenn Sie sicher sind, dass

die Standardwerte die gewünschten Werte sind oder nicht geändert wurden. Weitere Informationen hierzu finden Sie im Abschnitt „Attribute des Verwaltungsthemenobjekts anzeigen” auf Seite 95.

- Wenn Sie bereits ein Verwaltungsthema auf demselben WS-Manager mit dem Namen ORANGE.TOPIC haben, schlägt dieser Befehl fehl.Verwenden Sie das Attribut REPLACE, wenn Sie die vorhandene Definition eines Themas überschreiben möchten. Lesen Sie dazu auch „Verwaltungsthemenattribute ändern” auf Seite 95

Attribute des Verwaltungsthemenobjekts anzeigen

Mit dem MQSC-Befehl **DISPLAY TOPIC** können Sie ein Verwaltungsthemenobjekt anzeigen.

Um alle Themen anzuzeigen, verwenden Sie:

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

Sie können Attribute selektiv anzeigen, indem Sie sie einzeln angeben.Beispiel:

```
DISPLAY TOPIC(ORANGE.TOPIC) +
        TOPICSTR +
        DEFPRTY +
        NPMSGDLV
```

Mit diesem Befehl werden die drei angegebenen Attribute wie folgt angezeigt:

```
AMQ8633: Display topic details.
   TOPIC(ORANGE.TOPIC)                     TYPE(LOCAL)
   TOPICSTR(ORANGE)                        DEFPRTY(ASPARENT)
   NPMSGDLV(ASPARENT)
```

Verwenden Sie den Befehl DISPLAY TPSTATUS, um die ASPARENT-Werte für das Thema anzuzeigen, während sie in der Laufzeit verwendet werden. Verwenden Sie zum Beispiel:

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

Der Befehl zeigt die folgenden Details an:

```
AMQ8754: Display topic status details.
   TOPICSTR(ORANGE)                        DEFPRTY(0)
   NPMSGDLV(ALLAVAIL)
```

Wenn Sie ein Verwaltungsthema definieren, werden alle Attribute, die Sie nicht explizit angeben, aus dem Standardverwaltungsthema, das als SYSTEM.DEFAULT.TOPIC. bezeichnet wird, verwendet. Verwenden Sie den folgenden Befehl, um zu sehen, welche Standardattribute verwendet werden:

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

Verwaltungsthemenattribute ändern

Sie können Themenattribute auf zwei Arten ändern: mit dem Befehl **ALTER TOPIC** oder mit dem Befehl **DEFINE TOPIC** und dem Attribut **REPLACE** .

Wenn Sie beispielsweise die Standardpriorität von Nachrichten ändern möchten, die an ein Thema mit dem Namen ORANGE.TOPIC übergeben werden, müssen Sie einen der folgenden Befehle verwenden.

- Mit dem Befehl **ALTER** :

  ```
  ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
  ```

  Dieser Befehl ändert ein einzelnes Attribut, das der Standardpriorität der Nachricht, die diesem Thema zugestellt wird, auf 5; alle anderen Attribute bleiben unverändert.

- Mit dem Befehl **DEFINE** :

  ```
  DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
  ```

Mit diesem Befehl wird die Standardpriorität von Nachrichten geändert, die zu diesem Thema bereitgestellt werden. Alle anderen Attribute werden mit ihren Standardwerten angegeben.

Wenn Sie die Priorität von Nachrichten ändern, die an dieses Thema gesendet werden, sind die vorhandenen Nachrichten nicht betroffen. Jede neue Nachricht verwendet jedoch die angegebene Priorität, wenn sie nicht von der Veröffentlichungsanwendung bereitgestellt wird.

Eine Verwaltungsthemendefinition kopieren

Sie können eine Themendefinition mit dem Attribut LIKE im Befehl **DEFINE** kopieren.

Beispiel:

```
DEFINE TOPIC (MAGENTA.TOPIC) +
       LIKE (ORANGE.TOPIC)
```

Dieser Befehl erstellt ein Thema, MAGENTA.TOPIC, mit denselben Attributen wie das ursprüngliche Thema ORANGE.TOPIC und nicht die Attribute des Standardverwaltungsthemas für das System. Geben Sie den Namen des Themas ein, das genau so kopiert werden soll, wie es bei der Erstellung des Themas eingegeben wurde. Wenn der Name Kleinbuchstaben enthält, schließen Sie den Namen in einfache Anführungszeichen ein.

Sie können diese Form des Befehls **DEFINE** auch verwenden, um eine Themendefinition zu kopieren, aber Änderungen an den Attributen des Originals vorzunehmen. Beispiel:

```
DEFINE TOPIC(BLUE.TOPIC) +
       TOPICSTR(BLUE) +
       LIKE(ORANGE.TOPIC)
```

Sie können auch die Attribute des Themas BLUE.TOPIC in das Thema "GREEN.TOPIC" kopieren und angeben, dass, wenn Veröffentlichungen nicht an die richtige Subskribentenwarteschlange zugestellt werden können, sie nicht in die Warteschlange für dead-letter gestellt werden. Beispiel:

```
DEFINE TOPIC(GREEN.TOPIC) +
       TOPICSTR(GREEN) +
       LIKE(BLUE.TOPIC) +
       USEDLQ(NO)
```

Löschen einer Verwaltungsthemendefinition

Mit dem MQSC-Befehl **DELETE TOPIC** können Sie ein Verwaltungsthema löschen.

```
DELETE TOPIC(ORANGE.TOPIC)
```

Anwendungen können das Thema nicht mehr für die Veröffentlichung öffnen oder neue Subskriptionen unter Verwendung des Objektnamens ORANGE.TOPIC. öffnen.Die Veröffentlichung von Anwendungen, für die das Thema geöffnet ist, kann die Veröffentlichung der aufgelösten Themenzeichenfolge fortsetzen.Alle Subskriptionen, die bereits zu diesem Thema ausgeführt wurden, erhalten weiterhin Veröffentlichungen, nachdem das Thema gelöscht wurde.

Anwendungen, die dieses Themenobjekt nicht referenzieren, sondern die aufgelöste Themenzeichenfolge verwenden, die dieses Themenobjekt (in diesem Beispiel 'ORANGE') darstellt, arbeiten weiter. In diesem Fall übernehmen sie die Eigenschaften aus einem Themenobjekt in der Themenstruktur. Weitere Informationen zu Themenstrukturen finden Sie unter Themenstrukturen.

Mit Subskriptionen arbeiten

Verwenden Sie MQSC-Befehle zum Verwalten von Subskriptionen.

Subskriptionen können einen von drei Typen haben, die im Attribut **SUBTYPE** definiert sind:

ADMIN
Administrativ definiert durch einen Benutzer.

PROXY
Eine intern erstellte Subskription für Routing-Veröffentlichungen zwischen WS-Managern.

API
Das Programm wurde über das Programm erstellt, z. B. mit dem MQI-Aufruf MQSUB.

Ausführliche Informationen zu diesen Befehlen finden Sie im WebSphere MQ-Scriptbefehlsreferenz (MQSC) .

Zugehörige Konzepte

„Verwaltungssubskription definieren” auf Seite 97
Mit dem MQSC-Befehl **DEFINE SUB** können Sie eine Verwaltungssubskription erstellen. Sie können auch den Standardwert verwenden, der in der standardmäßigen lokalen Subskriptionsdefinition definiert ist. Sie können auch die Subskriptionsmerkmale von denen der lokalen Standardsubskription SYSTEM.DEFAULT.SUB ändern, die bei der Installation des Systems erstellt wurde.

„Attribute von Subskriptionen anzeigen” auf Seite 98
Mit dem Befehl **DISPLAY SUB** können Sie konfigurierte Attribute jeder Subskription anzeigen, die dem Warteschlangenmanager bekannt ist.

„Attribute für lokale Subskription ändern” auf Seite 99
Sie können Subskriptionsattribute auf zwei Arten ändern: mit dem Befehl **ALTER SUB** oder dem Befehl **DEFINE SUB** mit dem Attribut **REPLACE** .

„Lokale Subskriptionsdefinition kopieren” auf Seite 99
Sie können eine Subskriptionsdefinition mit dem Attribut **LIKE** im Befehl **DEFINE** kopieren.

„Subskription löschen” auf Seite 99
Mit dem MQSC-Befehl **DELETE SUB** können Sie eine lokale Subskription löschen.

Verwaltungssubskription definieren

Mit dem MQSC-Befehl **DEFINE SUB** können Sie eine Verwaltungssubskription erstellen. Sie können auch den Standardwert verwenden, der in der standardmäßigen lokalen Subskriptionsdefinition definiert ist. Sie können auch die Subskriptionsmerkmale von denen der lokalen Standardsubskription SYSTEM.DEFAULT.SUB ändern, die bei der Installation des Systems erstellt wurde.

Der folgende Befehl **DEFINE SUB** definiert beispielsweise eine Subskription mit dem Namen ORANGE mit den folgenden Merkmalen:

- Dauerhafte Subskription, d. d. sie bleibt beim Neustart des Warteschlangenmanagers bestehen, mit unbegrenztem Ablaufdatum.
- Empfangen von Veröffentlichungen, die in der Themenzeichenfolge ORANGE erfolgen, mit den Nachrichtenprioritäten, die von den Veröffentlichungsanwendungen festgelegt werden.
- Veröffentlichungen, die für diese Subskription bereitgestellt werden, werden an die lokale Warteschlange SUBQ gesendet. Diese Warteschlange muss vor der Definition der Subskription definiert werden.

```
DEFINE SUB (ORANGE) +
       TOPICSTR (ORANGE) +
       DESTCLAS (PROVIDED) +
       DEST (SUBQ) +
       EXPIRY (UNLIMITED) +
       PUBPRTY (ASPUB)
```

Anmerkung:

- Die Subskription und der Name der Themenzeichenfolge müssen nicht übereinstimmen.
- Mit Ausnahme der Werte der Beschreibung und der Themenzeichenfolge sind alle angezeigten Attributwerte die Standardwerte.Sie sind hier nur als Illustration dargestellt.Sie können sie übergehen, wenn Sie sicher sind, dass die Standardwerte die gewünschten Werte sind oder nicht geändert wurden. Weitere Informationen hierzu finden Sie im Abschnitt „Attribute von Subskriptionen anzeigen” auf Seite 98.

- Wenn Sie bereits über eine lokale Subskription auf demselben Warteschlangenmanager mit dem Namen TEST verfügen, schlägt dieser Befehl fehl.Verwenden Sie das Attribut **REPLACE**, wenn die vorhandene Warteschlangendefinition überschrieben werden soll. Lesen Sie dazu auch „Attribute für lokale Subskription ändern” auf Seite 99.
- Wenn die Warteschlange SUBQ nicht vorhanden ist, schlägt dieser Befehl fehl.

Attribute von Subskriptionen anzeigen

Mit dem Befehl **DISPLAY SUB** können Sie konfigurierte Attribute jeder Subskription anzeigen, die dem Warteschlangenmanager bekannt ist.

Verwenden Sie zum Beispiel:

```
DISPLAY SUB (ORANGE)
```

Sie können Attribute selektiv anzeigen, indem Sie sie einzeln angeben. Beispiel:

```
DISPLAY SUB (ORANGE) +
        SUBID +
        TOPICSTR +
        DURABLE
```

Mit diesem Befehl werden die drei angegebenen Attribute wie folgt angezeigt:

```
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

TOPICSTR ist die aufgelöste Themenzeichenfolge, auf der dieser Subskribent ausgeführt wird.Wenn eine Subskription definiert ist, um ein Topic-Objekt zu verwenden, wird die Themenzeichenfolge aus diesem Objekt als Präfix für die Themenzeichenfolge verwendet, die beim Herstellen der Subskription angegeben wird. SUBID ist eine eindeutige Kennung, die vom WS-Manager bei der Erstellung einer Subskription zugeordnet wird.Dies ist ein nützliches Attribut, das angezeigt werden kann, weil einige Subskriptionsnamen lang sein können oder in einem anderen Zeichensatz enthalten sind, für den sie möglicherweise unpraktisch werden.

Eine alternative Methode zum Anzeigen von Subskriptionen ist die Verwendung der SUBID:

```
DISPLAY SUB +
        SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
        TOPICSTR +
        DURABLE
```

Mit diesem Befehl wird die gleiche Ausgabe wie zuvor ausgegeben:

```
AMQ8096: WebSphere MQ subscription inquired.
 SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

Proxy-Subskriptionen in einem Warteschlangenmanager werden standardmäßig nicht angezeigt. Um sie anzuzeigen, geben Sie eine **SUBTYPE** von PROXY oder ALL an.

Sie können den Befehl DISPLAY SBSTATUS verwenden, um die Laufzeitattribute anzuzeigen. Verwenden Sie z. B. den folgenden Befehl:

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

Die folgende Ausgabe wird angezeigt:

```
AMQ8099: WebSphere MQ subscription status inquired.
   SUB(ORANGE)
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   NUMMSGS(0)
```

Wenn Sie eine Verwaltungssubskription definieren, werden alle Attribute, die Sie nicht explizit angeben, aus der Standardsubskription, die als SYSTEM.DEFAULT.SUB bezeichnet wird, verwendet. Verwenden Sie den folgenden Befehl, um zu sehen, welche Standardattribute verwendet werden:

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

Attribute für lokale Subskription ändern

Sie können Subskriptionsattribute auf zwei Arten ändern: mit dem Befehl **ALTER SUB** oder dem Befehl **DEFINE SUB** mit dem Attribut **REPLACE** .

Wenn Sie beispielsweise die Priorität von Nachrichten, die an eine Subskription mit dem Namen ORANGE übergeben werden, ändern möchten, verwenden Sie einen der folgenden Befehle:

- Befehl ALTER verwenden:

  ```
  ALTER SUB(ORANGE) PUBPRTY(5)
  ```

  Dieser Befehl ändert ein einzelnes Attribut, das der Priorität von Nachrichten, die an diese Subskription zugestellt werden, auf 5; alle anderen Attribute bleiben unverändert.
- Befehl DEFINE verwenden:

  ```
  DEFINE SUB (ORANGE) PUBPRTY(5) REPLACE
  ```

  Dieser Befehl ändert nicht nur die Priorität von Nachrichten, die an diese Subskription übergeben werden, sondern auch alle anderen Attribute, die ihre Standardwerte erhalten.

Wenn Sie die Priorität von Nachrichten ändern, die an diese Subskription gesendet werden, sind die vorhandenen Nachrichten nicht betroffen.Alle neuen Nachrichten haben jedoch die angegebene Priorität.

Lokale Subskriptionsdefinition kopieren

Sie können eine Subskriptionsdefinition mit dem Attribut **LIKE** im Befehl **DEFINE** kopieren.

Beispiel:

```
DEFINE SUB (BLUE) +
       LIKE (ORANGE)
```

Sie können auch die Attribute der untergeordneten REAL in das untergeordnete Element THIRD.SUB kopieren und angeben, dass die Korrelations-ID der bereitgestellten Veröffentlichungen DIRD ist und nicht die Publisher-CorrelID. Beispiel:

```
DEFINE SUB(THIRD.SUB) +
       LIKE(BLUE) +
       DESTCORL(ORANGE)
```

Subskription löschen

Mit dem MQSC-Befehl **DELETE SUB** können Sie eine lokale Subskription löschen.

```
DELETE SUB(ORANGE)
```

Sie können eine Subskription auch mit der SUBID löschen:

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

Nachrichten in einer Subskription überprüfen

Informationen zu diesem Vorgang

Wenn eine Subskription definiert ist, wird sie einer Warteschlange zugeordnet. Veröffentlichte Nachrichten, die mit dieser Subskription übereinstimmen, werden in diese Warteschlange gestellt.

Beachten Sie, dass die folgenden **runmqsc** -Befehle nur die Subskriptionen anzeigen, die Nachrichten erhalten haben.

Führen Sie die folgenden Schritte aus, um nach Nachrichten zu suchen, die sich derzeit in der Warteschlange für eine Subskription befinden

Vorgehensweise

1. Informationen zum Überprüfen von Nachrichten, die sich in der Warteschlange für einen Subskriptionstyp **DISPLAY SBSTATUS(<sub_name>) NUMMSGS** befinden, finden Sie im Abschnitt „Attribute von Subskriptionen anzeigen" auf Seite 98.
2. Wenn der Wert für **NUMMSGS** größer als null ist, identifizieren Sie die Warteschlange, die der Subskription zugeordnet ist, indem Sie **DISPLAY SUB(<sub_name>)DEST** eingeben.
3. Wenn Sie den Namen der zurückgegebenen Warteschlange verwenden, können Sie die Nachrichten nach dem in „Warteschlangen durchsuchen" auf Seite 89 beschriebenen Verfahren anzeigen.

Mit Services arbeiten

Serviceobjekte sind ein Mittel, mit dem zusätzliche Prozesse als Teil eines Warteschlangenmanagers verwaltet werden können. Mit Services können Sie Programme definieren, die gestartet und gestoppt werden, wenn der WS-Manager gestartet und beendet wird. IBM WebSphere MQ-Services werden immer unter der Benutzer-ID des Benutzers gestartet, der den Warteschlangenmanager gestartet hat.

Serviceobjekte können einen der folgenden Typen haben:

Server
Ein Server ist ein Serviceobjekt, dessen Parameter SERVTYPE auf SERVER gesetzt ist. Ein Serverserviceobjekt ist die Definition eines Programms, das ausgeführt wird, wenn ein bestimmter WS-Manager gestartet wird. Serverserviceobjekte definieren Programme, die in der Regel lange ausgeführt werden. Ein Server-Service-Objekt kann beispielsweise verwendet werden, um einen Auslösemonitorprozess wie `runmqtrm`auszuführen.

Es kann nur eine Instanz eines Server-Service-Objekts gleichzeitig ausgeführt werden. Der Status aktiver Serverserviceobjekte kann mit dem MQSC-Befehl DISPLAY SVSTATUS überwacht werden.

Befehl
Ein Befehl ist ein Serviceobjekt, dessen Parameter SERVTYPE auf COMMAND gesetzt ist. Befehlserviceobjekte sind mit Serverserviceobjekten vergleichbar. Es können jedoch mehrere Instanzen eines Befehlserviceobjekts gleichzeitig ausgeführt werden und ihr Status lässt sich nicht mit dem MQSC-Befehl DISPLAY SVSTATUS überwachen.

Bei Verwendung des MQSC-Befehls STOP SERVICE wird vor Ausführung des Stoppprogramms nicht geprüft, ob das mit dem MQSC-Befehl START SERVICE gestartete Programm noch aktiv ist.

Serviceobjekt definieren

Sie können ein Serviceobjekt mit verschiedenen Attributen definieren.

Folgende Attribute werden verwendet:

SERVTYPE
Definiert den Typ des Serviceobjekts. Folgende Werte sind möglich:

Server
Ein Serverserviceobjekt.

Es kann immer nur eine Instanz eines Serverserviceobjekts ausgeführt werden. Der Status eines Serverserviceobjekts kann mit dem MQSC-Befehl DISPLAY SVSTATUS überwacht werden.

Befehl
Ein Befehlsserviceobjekt.

Es können mehrere Instanzen eines Befehlsserviceobjekts gleichzeitig ausgeführt werden. Der Status einer Befehlsserviceobjekte kann nicht überwacht werden.

STARTCMD
Das Programm, das zum Starten des Service ausgeführt wird. Es muss ein vollständig qualifizierter Pfad zum Programm angegeben werden.

STARTARG
Argumente, die an das Startprogramm übergeben werden.

STDERR
Gibt den Pfad zu einer Datei an, in die der Standardfehler (stderr) des Serviceprogramms umgeleitet werden soll.

STDOUT
Gibt den Pfad zu einer Datei an, in die die Standardausgabe (stdout) des Serviceprogramms umgeleitet werden soll.

STOPCMD
Das Programm, das zum Stoppen des Service ausgeführt wird. Es muss ein vollständig qualifizierter Pfad zum Programm angegeben werden.

STOPARG
Argumente, die an das Stoppprogramm übergeben werden.

STEUERUNG
Gibt an, wie der Service gestartet und gestoppt werden soll:

MANUAL
Der Service wird nicht automatisch gestartet oder automatisch gestoppt. Er wird durch die Verwendung der Befehle START SERVICE und STOP SERVICE gesteuert. Dies ist der Standardwert.

QMGR
Der Service, der definiert wird, soll gleichzeitig gestartet und gestoppt werden, wenn der Warteschlangenmanager gestartet und gestoppt wird.

STARTONLY
Der Service soll zur gleichen Zeit wie der Warteschlangenmanager gestartet werden, aber er wird nicht zum Stoppen aufgefordert, wenn der Warteschlangenmanager gestoppt wird.

Zugehörige Konzepte
„Services verwalten” auf Seite 101
Wenn Sie den Parameter CONTROL verwenden, kann eine Instanz eines Serviceobjekts entweder automatisch vom Warteschlangenmanager gestartet oder gestoppt werden oder mit den MQSC-Befehlen START SERVICE und STOP SERVICE gestartet und gestoppt werden.

Services verwalten

Wenn Sie den Parameter CONTROL verwenden, kann eine Instanz eines Serviceobjekts entweder automatisch vom Warteschlangenmanager gestartet oder gestoppt werden oder mit den MQSC-Befehlen START SERVICE und STOP SERVICE gestartet und gestoppt werden.

Wenn eine Instanz eines Serviceobjekts gestartet wird, wird eine Nachricht in das Fehlerprotokoll des Warteschlangenmanagers geschrieben, in der der Name des Serviceobjekts und die Prozess-ID des gestarteten Prozesses enthalten sind. Es folgt ein Beispielprotokolleintrag für ein Serverserviceobjekt:

```
  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.
```

Es folgt ein Beispielprotokolleintrag für ein Befehlsserviceobjekt:

```
  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.
```

Wenn ein Instanzserver-Service gestoppt wird, wird eine Nachricht in die Fehlerprotokolle des Warteschlangenmanagers geschrieben, die den Namen des Service und die Prozess-ID des Endprozesses enthalten. Im Folgenden wird ein Beispielprotokolleintrag für ein Serverserviceobjekt gestoppt:

```
  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.
```

Zugehörige Verweise

„Zusätzliche Umgebungsvariablen” auf Seite 102
Wenn ein Service gestartet wird, wird die Umgebung, in der der Serviceprozess gestartet wird, aus der Umgebung des Warteschlangenmanagers übernommen. Sie können zusätzliche Umgebungsvariablen definieren, die in der Umgebung des Serviceprozesses festgelegt werden sollen, indem Sie die Variablen, die Sie definieren möchten, zu einer der `service.env`-Umgebungsüberschreibungsdateien hinzufügen.

Zusätzliche Umgebungsvariablen

Wenn ein Service gestartet wird, wird die Umgebung, in der der Serviceprozess gestartet wird, aus der Umgebung des Warteschlangenmanagers übernommen. Sie können zusätzliche Umgebungsvariablen definieren, die in der Umgebung des Serviceprozesses festgelegt werden sollen, indem Sie die Variablen, die Sie definieren möchten, zu einer der `service.env`-Umgebungsüberschreibungsdateien hinzufügen.

Anmerkung:

Es gibt zwei mögliche Dateien, denen Sie Umgebungsvariablen hinzufügen können:

- Die Datei `service.env` im Maschinengeltungsbereich, die sich in `/var/mqm` auf UNIX and Linux -Systemen oder in dem Datenverzeichnis befindet, das bei der Installation auf Windows -Systemen ausgewählt wurde
- Die Datei mit dem Warteschlangenmanagerbereich `service.env`, die sich im Datenverzeichnis des Warteschlangenmanagers befindet. Beispiel: Die Position der Umgebungsüberschreibungsdatei für einen WS-Manager mit dem Namen `QMNAME` lautet:
  - Auf Systemen unter UNIX and Linux, `/var/mqm/qmgrs/QMNAME/service.env`
  - Auf Systemen unter Windows, `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`

Beide Dateien werden, falls verfügbar, verarbeitet, wobei Definitionen in der Bereichsdatei des Warteschlangenmanagers Vorrang vor diesen Definitionen in der Datei des Maschinengeltungsbereichs haben.

Jede Umgebungsvariable kann in `service.env`angegeben werden. Wenn der IBM WebSphere MQ -Service beispielsweise eine Reihe von Befehlen ausführt, kann es sinnvoll sein, die Variable PATH in der Datei `service.env` zu definieren. Die Werte, die Sie für die Variable festlegen, können keine Umgebungsvari-

ablen sein. Beispiel: CLASSPATH=*%CLASSPATH%* ist falsch. In ähnlicher Weise würde `:/opt/mqm/bin` unter Linux PATH=*$PATH*unerwartete Ergebnisse liefern.

CLASSPATH muss in Großbuchstaben geschrieben werden, und die Klassenpfadanweisung darf nur Literale enthalten. Einige Services (z. B. Telemetrie) legen ihren eigenen Klassenpfad fest. Der in `service.env` definierte CLASSPATH wird diesem hinzugefügt.

Das Format der in der Datei definierten Variablen`service.env` ist eine Liste von Name/Wert-Variablenpaaren. Jede Variable muss in einer neuen Zeile definiert werden, und jede Variable wird so verwendet, wie sie explizit definiert ist, einschließlich des Leerraums. Es folgt ein Beispiel für die Datei`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
```

Zugehörige Verweise

„Ersetzbare Einfügungen in Servicedefinitionen” auf Seite 103
In der Definition eines Serviceobjekts ist es möglich, Token zu ersetzen. Ersetzte Token werden automatisch durch ihren erweiterten Text ersetzt, wenn das Serviceprogramm ausgeführt wird. Ersatztoken können aus der folgenden Liste der allgemeinen Token oder aus allen Variablen, die in der Datei service.env definiert sind, entnommen werden.

Ersetzbare Einfügungen in Servicedefinitionen

In der Definition eines Serviceobjekts ist es möglich, Token zu ersetzen. Ersetzte Token werden automatisch durch ihren erweiterten Text ersetzt, wenn das Serviceprogramm ausgeführt wird. Ersatztoken können aus der folgenden Liste der allgemeinen Token oder aus allen Variablen, die in der Datei service.env definiert sind, entnommen werden.

Im Folgenden finden Sie allgemeine Tokens, die verwendet werden können, um Token in der Definition eines Serviceobjekts zu ersetzen:

MQ_INSTALLATIONSPFAD
Die Position, an der WebSphere MQ installiert ist.

MQ_DATA_PATH
Die Position des WebSphere MQ -Datenverzeichnisses:

- Auf UNIX and Linux -Systemen lautet die Position des Datenverzeichnisses für WebSphere MQ `/var/mqm/`
- Auf Windows -Systemen ist die Position des WebSphere MQ -Datenverzeichnisses das bei der Installation von WebSphere MQ ausgewählte Datenverzeichnis.

QMNAME
Der Name des aktuellen Warteschlangenmanagers.

MQ_SERVICE_NAME
Der Name des Service.

MQ_SERVER_PID
Dieses Token kann nur von den Argumenten STOPARG und STOPCMD verwendet werden.

Für Serverserviceobjekte wird dieses Token durch die Prozess-ID des Prozesses ersetzt, der von den Argumenten STARTCMD und STARTARG gestartet wird. Andernfalls wird dieses Token durch 0 ersetzt.

MQ_Q_MGR_DATA_PATH
Die Position des Datenverzeichnisses des Warteschlangenmanagers.

MQ_Q_MGR_DATA_NAME
Der umgewandelte Name des Warteschlangenmanagers. Weitere Informationen zur Namensumsetzung finden Sie unter Understanding WebSphere MQ file names.

Wenn Sie austauschbare Einfügungen verwenden möchten, fügen Sie das Token in +-Zeichen in die Zeichenfolgen STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT oder STDERR ein. Beispiele hierzu finden Sie im Abschnitt „Beispiele zur Verwendung von Serviceobjekten” auf Seite 104.

Beispiele zur Verwendung von Serviceobjekten

Die Services in diesem Abschnitt werden mit UNIX -Pfadtrennzeichen geschrieben, sofern nicht anders angegeben.

Serverserviceobjekt verwenden

In diesem Beispiel wird gezeigt, wie ein Serverserviceobjekt definiert, verwendet und geändert wird, um einen Auslösemonitor zu starten.

1. Ein Serverserviceobjekt wird mit dem folgenden MQSC-Befehl definiert:

```
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+')
```

   Dabei gilt Folgendes:

   +MQ_INSTALL_PATH+ ist ein Token, das das Installationsverzeichnis darstellt.
   +QMNAME+ ist ein Token, das den Namen des Warteschlangenmanagers darstellt.
   ACCOUNTS.INITIATION.QUEUE ist die Initialisierungswarteschlange.
   amqsstop ist ein Beispielprogramm, das mit WebSphere MQ bereitgestellt wird und den Warteschlangenmanager auffordert, alle Verbindungen für die Prozess-ID zu unterbrechen. amqsstop generiert PCF-Befehle, daher muss der Befehlsserver aktiv sein.
   +MQ_SERVER_PID+ ist ein Token, das die Prozess-ID darstellt, die an das Stoppprogramm übergeben wurde.

   Eine Liste der allgemeinen Tokens finden Sie im Abschnitt „Ersetzbare Einfügungen in Servicedefinitionen” auf Seite 103.
2. Eine Instanz des Serverserviceobjekts wird ausgeführt, wenn der WS-Manager als Nächstes gestartet wird. Es wird jedoch sofort eine Instanz des Serverserviceobjekts mit dem folgenden MQSC-Befehl gestartet:

```
START SERVICE(S1)
```

3. Der Status des Serverserviceprozesses wird mit dem folgenden MQSC-Befehl angezeigt:

```
DISPLAY SVSTATUS(S1)
```

4. In diesem Beispiel wird jetzt gezeigt, wie das Serverserviceobjekt geändert wird und dass die Aktualisierungen durch einen manuellen Neustart des Serverserviceprozesses übernommen werden. Das Serverserviceobjekt wird so geändert, dass die Initialisierungswarteschlange als JUPITER.INITIATION.QUEUE angegeben wird. Der folgende MQSC-Befehl wird verwendet:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

Anmerkung: Ein laufender Service nimmt keine Aktualisierungen an seiner Servicedefinition an, bis er erneut gestartet wird.

5. Der Serverserviceprozess wird erneut gestartet, so dass die Änderungen mit den folgenden MQSC-Befehlen erfasst werden:

```
STOP SERVICE(S1)
```

Gefolgt von:

```
START SERVICE(S1)
```

Der Serverserviceprozess wird erneut gestartet und nimmt die in „4” auf Seite 105 vorgenommenen Änderungen wieder auf.

Anmerkung: Der MQSC-Befehl STOP SERVICE kann nur verwendet werden, wenn ein STOPCMD-Argument in der Servicedefinition angegeben ist.

Befehlsserviceobjekt verwenden

In diesem Beispiel wird gezeigt, wie ein Befehlsserviceobjekt definiert wird, um ein Programm zu starten, das Einträge in das Systemprotokoll des Betriebssystems schreibt, wenn ein Warteschlangenmanager gestartet oder gestoppt wird.

1. Das Befehlsserviceobjekt wird mit dem folgenden MQSC-Befehl definiert:

```
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')
```

Dabei gilt Folgendes:

logger ist der von UNIX and Linux bereitgestellte Befehl zum Schreiben in das Systemprotokoll.
+QMNAME+ ist ein Token, das den Namen des Warteschlangenmanagers darstellt.

Verwenden eines Befehls-Service-Objekts, wenn ein Warteschlangenmanager nur beendet wird

Dieses Beispiel zeigt, wie ein Befehlsserviceobjekt definiert wird, um ein Programm zu starten, das Einträge in das Systemprotokoll des Betriebssystems schreibt, wenn ein WS-Manager nur gestoppt wird.

1. Das Befehlsserviceobjekt wird mit dem folgenden MQSC-Befehl definiert:

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Dabei gilt Folgendes:

`logger` ist ein Beispielprogramm, das mit WebSphere MQ bereitgestellt wird und Einträge in das Systemprotokoll des Betriebssystems schreiben kann.

+QMNAME+ ist ein Token, das den Namen des Warteschlangenmanagers darstellt.

Weitere Informationen zum Übergeben von Argum

In diesem Beispiel wird gezeigt, wie ein Serverserviceobjekt definiert wird, um ein Programm mit dem Namen `runserv` zu starten, wenn ein Warteschlangenmanager gestartet wird.

Dieses Beispiel wird mit Windows -Pfadtrennzeichen geschrieben.

Eines der Argumente, die an das Startprogramm übergeben werden sollen, ist eine Zeichenfolge, die einen Speicherbereich enthält. Dieses Argument muss als einzelne Zeichenfolge übergeben werden. Um dies zu erreichen, werden doppelte Anführungszeichen verwendet, wie im folgenden Befehl gezeigt, um das Befehlsserviceobjekt zu definieren:

1. Das Serverserviceobjekt wird mit dem folgenden MQSC-Befehl definiert:

```
   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')
```

Dabei gilt Folgendes:

+QMNAME+ ist ein Token, das den Namen des Warteschlangenmanagers darstellt.

`"C:\Program Files\Tools\"` ist eine Zeichenfolge mit einem Leerzeichen, die als einzelne Zeichenfolge übergeben wird.

Automatisches Starten eines Service

Dieses Beispiel zeigt, wie ein Serverserviceobjekt definiert wird, das zum automatischen Starten des Auslösemonitors beim Starten des Warteschlangenmanagers verwendet werden kann.

1. Das Serverserviceobjekt wird mit dem folgenden MQSC-Befehl definiert:

```
DEFINE SERVICE(TRIG_MON_START) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('runmqtrm') +
       STARTARG('-m +QMNAME+ -q +IQNAME+')
```

Dabei gilt Folgendes:

+QMNAME+ ist ein Token, das den Namen des Warteschlangenmanagers darstellt.

+IQNAME+ ist eine vom Benutzer in einer der service.env-Dateien definierte Umgebungsvariable, die den Namen der Initialisierungswarteschlange darstellt.

Objekte zum Auslösen verwalten

WebSphere MQ ermöglicht Ihnen, eine Anwendung automatisch zu starten, wenn bestimmte Bedingungen in einer Warteschlange erfüllt sind. Beispiel: Sie möchten eine Anwendung starten, wenn die Anzahl der Nachrichten in einer Warteschlange eine angegebene Zahl erreicht. Diese Funktion wird als *Auslösefunktion* bezeichnet. Sie müssen die Objekte definieren, die das Auslösen unterstützen.

Die Auslösefunktion wird ausführlich im Abschnitt WebSphere MQ -Anwendungen mit Triggern startenbeschrieben.

Anwendungswarteschlange zum Auslösen definieren

Eine Anwendungswarteschlange ist eine lokale Warteschlange, die von Anwendungen für die Nachrichtenübertragung über die MQI verwendet wird. Für die Triggerung ist es erforderlich, dass eine Reihe von Warteschlangenattributen in der Anwendungswarteschlange definiert wird.

Die Triggerung selbst wird durch das Attribut *Trigger* (TRIGGER in MQSC-Befehlen) aktiviert. In diesem Beispiel soll ein Auslöserereignis generiert werden, wenn 100 Nachrichten mit Priorität 5 oder höher in der lokalen Warteschlange MOTOR.INSURANCE.QUEUE vorhanden sind, wie folgt:

```
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)
```

Dabei gilt:

QLOCAL (MOTOR.INSURANCE.QUEUE)
Ist der Name der Anwendungswarteschlange, die definiert wird.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
Ist der Name der Prozessdefinition, die die Anwendung definiert, die von einem Auslösemonitorprogramm gestartet werden soll.

MAXMSGL (2000)
Gibt die maximale Länge von Nachrichten in der Warteschlange an.

DEFPSIST (YES)
Gibt an, dass Nachrichten in dieser Warteschlange standardmäßig persistent sind.

INITQ (MOTOR.INS.INIT.QUEUE)
Gibt den Namen der Initialisierungswarteschlange an, in die der Warteschlangenmanager die Auslösenachricht stellen soll.

TRIGGER
Ist der Auslöserattributwert.

TRIGTYPE (DEPTH)
Gibt an, dass ein Auslöserereignis generiert wird, wenn die Anzahl der Nachrichten mit der erforderlichen Priorität (TRIGMPRI) die in TRIGDPTH angegebene Zahl erreicht.

TRIGDPTH (100)
Gibt die Anzahl der Nachrichten an, die zum Generieren eines Auslöserereignisses erforderlich sind.

TRIGMPRI (5)
Ist die Priorität von Nachrichten, die vom Warteschlangenmanager gezählt werden sollen, wenn Sie entscheiden, ob ein Auslöserereignis generiert werden soll. Es werden nur Nachrichten mit Priorität 5 oder höher gezählt.

Initialisierungswarteschlange definieren

Wenn ein Auslöserereignis eintritt, reiht der Warteschlangenmanager eine Auslösenachricht in die Initialisierungswarteschlange ein, die in der Definition der Anwendungswarteschlange angegeben ist. Initialisierungswarteschlangen haben keine speziellen Einstellungen. Sie können jedoch die folgende Definition der lokalen Warteschlange MOTOR.INS.INIT.QUEUE als Anleitung verwenden:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Prozess definieren

Verwenden Sie den Befehl DEFINE PROCESS, um eine Prozessdefinition zu erstellen. Eine Prozessdefinition definiert die Anwendung, die für die Verarbeitung von Nachrichten aus der Anwendungswarteschlange verwendet werden soll. Die Anwendungswarteschlangendefinition benennt den zu verwendenden Prozess und ordnet dadurch die Anwendungswarteschlange der Anwendung zu, die für die Verarbeitung ihrer Nachrichten verwendet werden soll. Dies wird über das Attribut PROCESS in der Anwendungswarteschlange MOTOR.INSURANCE.QUEUE ausgeführt. Der folgende MQSC-Befehl definiert den erforderlichen Prozess (MOTOR.INSURANCE.QUOTE.PROCESS), der in diesem Beispiel angegeben ist:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

Dabei gilt Folgendes:

MOTOR.INSURANCE.QUOTE.PROCESS
Ist der Name der Prozessdefinition.

DESCR ('Insurance request message processing')
Beschreibt das Anwendungsprogramm, auf das sich diese Definition bezieht. Dieser Text wird angezeigt, wenn Sie den Befehl DISPLAY PROCESS verwenden. Dadurch können Sie feststellen, was der Prozess tut. Wenn Sie Leerzeichen in der Zeichenfolge verwenden, müssen Sie die Zeichenfolge in einfache Anführungszeichen setzen.

APPLTYPE (UNIX)
Ist der Typ der Anwendung, die gestartet werden soll.

APPLICID ('/u/admin/test/IRMP01')
Ist der Name der ausführbaren Datei der Anwendung, die als vollständig qualifizierter Dateiname angegeben ist. Auf Windows -Systemen wäre ein typischer `APPLICID` -Wert `c:\appl\test\irmp01.exe`.

USERDATA ('open, close, 235')
Benutzerdefinierte Daten, die von der Anwendung verwendet werden können.

Attribute einer Prozessdefinition anzeigen

Verwenden Sie den Befehl DISPLAY PROCESS, um die Ergebnisse Ihrer Definition zu untersuchen. Beispiel:

```
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)
```

Sie können den MQSC-Befehl ALTER PROCESS auch verwenden, um eine vorhandene Prozessdefinition zu ändern, und den Befehl DELETE PROCESS, um eine Prozessdefinition zu löschen.

Ferne IBM WebSphere MQ -Objekte verwalten

In diesem Abschnitt erfahren Sie, wie Sie IBM WebSphere MQ-Objekte mithilfe von MQSC-Befehlen auf einem fernen Warteschlangenmanager verwalten und wie ferne Warteschlangenobjekte zum Steuern der Zieladresse von Nachrichten und Antwortnachrichten verwendet werden.

In diesem Abschnitt wird Folgendes beschrieben:

Kanäle, Cluster und Steuerung ferner Warteschlangen

Ein WS-Manager kommuniziert mit einem anderen Warteschlangenmanager, indem er eine Nachricht sendet und, falls erforderlich, eine Antwort zurückerhält. Der empfangende WS-Manager könnte wie folgt sein:

- Auf derselben Maschine
- Auf einer anderen Maschine in der gleichen Lage (oder sogar auf der anderen Seite der Welt)
- Ausführung auf derselben Plattform wie der lokale WS-Manager
- Ausführung auf einer anderen von WebSphere MQ unterstützten Plattform

Diese Nachrichten können aus folgenden Nachrichten stammen:

- Benutzergeschriebene Anwendungsprogramme, die Daten von einem Knoten in einen anderen übertragen
- Benutzergeschriebene Verwaltungsanwendungen, die PCF-Befehle oder die MQAI verwenden
- IBM WebSphere MQ Explorer
- Warteschlangenmanager senden:
  - Instrumentierungsereignisnachrichten an einen anderen WS-Manager
  - MQSC-Befehle, die von einem `runmqsc` -Befehl im indirekten Modus ausgegeben werden (wobei die Befehle auf einem anderen Warteschlangenmanager ausgeführt werden)

Bevor eine Nachricht an einen fernen Warteschlangenmanager gesendet werden kann, benötigt der lokale WS-Manager einen Mechanismus zum Erkennen der Ankunft von Nachrichten und Transport, die aus folgenden Bereichen bestehen:

- Mindestens ein Kanal
- eine Übertragungswarteschlange
- Kanalinitiator

Damit ein ferner Warteschlangenmanager eine Nachricht empfangen kann, ist ein Empfangsprogramm erforderlich.

Ein Kanal ist eine Einweg-Kommunikationsverbindung zwischen zwei Warteschlangenmanagern und kann Nachrichten übertragen, die für eine beliebige Anzahl von Warteschlangen auf dem fernen Warteschlangenmanager bestimmt sind.

Jedes Ende des Kanals verfügt über eine separate Definition. Wenn ein Ende z. B. ein Sender oder ein Server ist, muss das andere Ende ein Empfänger oder ein Requester sein. Ein einfacher Kanal besteht aus einer *Senderkanaldefinition* am lokalen WS-Manager-Ende und einer *Empfängerkanaldefinition* auf dem fernen WS-Manager-Ende. Die beiden Definitionen müssen denselben Namen haben und zusammen einen einzigen Nachrichtenkanal bilden.

Wenn der ferne WS-Manager auf Nachrichten antworten soll, die vom lokalen Warteschlangenmanager gesendet werden, konfigurieren Sie einen zweiten Kanal, um Antworten an den lokalen Warteschlangenmanager zu senden.

Verwenden Sie den MQSC-Befehl DEFINE CHANNEL, um Kanäle zu definieren. In diesem Abschnitt verwenden die Beispiele in Bezug auf Kanäle die Standardkanalattribute, sofern nicht anders angegeben.

An jedem Ende eines Kanals befindet sich ein Nachrichtenkanalagent (MCA), der das Senden und Empfangen von Nachrichten steuert. Der MCA nimmt Nachrichten aus der Übertragungswarteschlange und stellt sie in die Kommunikationsverbindung zwischen den Warteschlangenmanagern.

Eine Übertragungswarteschlange ist eine spezielle lokale Warteschlange, in der Nachrichten temporär gespeichert werden, bevor der MCA sie abholt und sie an den fernen Warteschlangenmanager sendet. Sie geben den Namen der Übertragungswarteschlange in einer *Definition der fernen Warteschlange* an.

Sie können einem Nachrichtenkanalagenten ermöglichen, Nachrichten unter Verwendung mehrerer Threads zu übertragen. Dieser Prozess ist als *Pipelining* bekannt. Mithilfe von Pipelining kann der Nachrichtenkanalagent Nachrichten effizienter übertragen, was die Kanalleistung verbessert. Ausführliche Informationen zum Konfigurieren eines Kanals für die Verwendung von Pipelining finden Sie unter Attribute von Kanälen .

In „Kanäle und Übertragungswarteschlangen für die Fernverwaltung vorbereiten” auf Seite 112 erfahren Sie, wie Sie diese Definitionen verwenden, um die Fernverwaltung einzurichten.

Weitere Informationen zum Einrichten der verteilten Warteschlangensteuerung im Allgemeinen finden Sie im Abschnitt Verteilte Warteschlangenkomponenten .

Fernverwaltung mit Clustern

In einem WebSphere MQ -Netz mit verteilter Steuerung von Warteschlangen ist jeder Warteschlangenmanager unabhängig. Wenn ein Warteschlangenmanager Nachrichten an einen anderen WS-Manager senden muss, muss er eine Übertragungswarteschlange, einen Kanal zum fernen Warteschlangenmanager und eine Definition einer fernen Warteschlange für jede Warteschlange definieren, an die Nachrichten gesendet werden sollen.

Ein *Cluster* ist eine Gruppe von Warteschlangenmanagern, die so konfiguriert sind, dass die Warteschlangenmanager über ein einziges Netz ohne komplexe Übertragungswarteschlange, Kanal und Warteschlangendefinitionen direkt miteinander kommunizieren können. Cluster können ohne großen Aufwand konfiguriert werden und enthalten in der Regel Warteschlangenmanager, die logisch miteinander verknüpft sind und Daten oder Anwendungen gemeinsam nutzen müssen. Selbst der kleinste Cluster reduziert die Systemverwaltungskosten.

Die Einrichtung eines Netzes von Warteschlangenmanagern in einem Cluster umfasst weniger Definitionen als die Erstellung einer traditionellen verteilten Warteschlangenumgebung. Wenn weniger Definitionen vorhanden sind, können Sie Ihr Netz schneller und einfacher einrichten oder ändern und das Risiko verringern, einen Fehler in Ihren Definitionen zu erstellen.

Um einen Cluster zu konfigurieren, benötigen Sie einen Clustersendersender (CLUSSDR) und eine Clusterempfängerdefinition (CLUSRCVR) für jeden Warteschlangenmanager. Sie benötigen keine Übertragungswarteschlangendefinitionen oder Definitionen ferner Warteschlangen. Die Prinzipien der Fernverwaltung sind dieselben, wenn sie in einem Cluster verwendet werden, aber die Definitionen selbst sind stark vereinfacht.

Weitere Informationen zu Clustern, ihren Attributen und ihrer Einrichtung finden Sie im Abschnitt Warteschlangenmanagercluster.

Ferne Verwaltung von einem lokalen WS-Manager

In diesem Abschnitt wird erläutert, wie ein ferner Warteschlangenmanager über MQSC- und PCF-Befehle von einem lokalen Warteschlangenmanager aus verwaltet werden kann.

Das Vorbereiten der Warteschlangen und Kanäle ist für die beiden MQSC-und PCF-Befehle im Wesentlichen identisch. In diesem Abschnitt werden in den Beispielen die MQSC-Befehle angezeigt, da sie einfacher zu verstehen sind. Weitere Informationen zum Schreiben von Verwaltungsprogrammen mit Hilfe von PCF-Befehlen finden Sie in „Programmierbare Befehlsformate verwenden” auf Seite 10.

Sie senden MQSC-Befehle entweder interaktiv oder aus einer Textdatei, die die Befehle enthält, an einen fernen Warteschlangenmanager. Der ferne WS-Manager kann sich auf derselben Maschine befinden oder, in der Regel, auf einer anderen Maschine. Sie können Warteschlangenmanager in anderen WebSphere

MQ -Umgebungen, einschließlich UNIX and Linux -Systemen, Windows -Systemen IBM iund z/OS, über Fernzugriff verwalten.

Um die Fernverwaltung zu implementieren, müssen Sie bestimmte Objekte erstellen. Sofern keine besonderen Anforderungen bestehen, können die Standardwerte (beispielsweise für die maximale Länge der Nachrichten) verwendet werden.

Vorbereiten von Warteschlangenmanagern für die Fernverwaltung

Verwendung von MQSC-Befehlen zum Vorbereiten von Warteschlangenmanagern für die Fernverwaltung.

Abbildung 17 auf Seite 111 zeigt die Konfiguration von Warteschlangenmanagern und Kanälen, die Sie für die Fernverwaltung mit dem Befehl **runmqsc** benötigen. Das Objekt `source.queue.manager` ist der Quellenwarteschlangenmanager, von dem aus Sie MQSC-Befehle ausgeben können und zu dem die Ergebnisse dieser Befehle (Bedienernachrichten) zurückgegeben werden. Das Objekt `target.queue.manager` ist der Name des Zielwarteschlangenmanagers, der die Befehle verarbeitet und alle Bedienernachrichten generiert.

Anmerkung: Wenn Sie **runmqsc** mit der Option '-w' verwenden, ***muss*** `source.queue.manager` der Standardwarteschlangenmanager sein. Weitere Informationen zum Erstellen eines Warteschlangenmanagers finden Sie unter crtmqm.

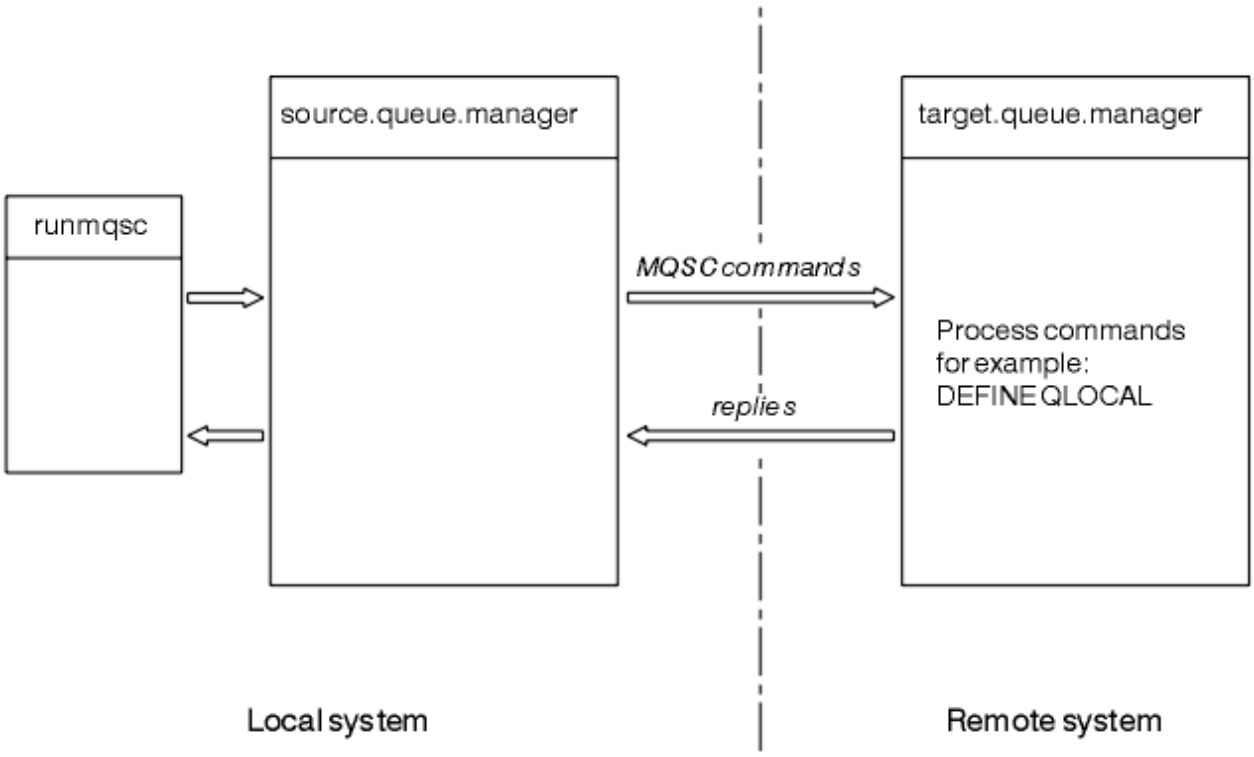


Abbildung 17. Ferne Verwaltung mit MQSC-Befehlen

Auf beiden Systemen, wenn Sie dies noch nicht getan haben:

- Erstellen Sie den Warteschlangenmanager und die Standardobjekte mit dem Befehl `crtmqm` .
- Starten Sie den Warteschlangenmanager mit dem Befehl `strmqm` .

Auf dem Zielwarteschlangenmanager:

- Die Befehlswarteschlange SYSTEM.ADMIN.COMMAND.QUEUE muss vorhanden sein. Diese Warteschlange wird standardmäßig bei der Erstellung eines Warteschlangenmanagers erstellt.

Sie müssen diese Befehle lokal oder über eine Netzfunktion wie Telnet ausführen.

Kanäle und Übertragungswarteschlangen für die Fernverwaltung vorbereiten

Verwendung von MQSC-Befehlen zum Bereiten von Kanälen und Übertragungswarteschlangen für die Fernverwaltung.

Um MQSC-Befehle über Remotezugriff auszuführen, konfigurieren Sie zwei Kanäle, eine für jede Richtung, und die zugehörigen Übertragungswarteschlangen. In diesem Beispiel wird vorausgesetzt, dass Sie TCP/IP als Transporttyp verwenden und dass Sie die TCP/IP-Adresse kennen.

Der Kanal `source.to.target` dient zum Senden von MQSC-Befehlen vom Quellenwarteschlangenmanager an den Ziel-WS-Manager. Der Sender befindet sich unter `source.queue.manager` und sein Empfänger befindet sich unter `target.queue.manager`. Der Kanal `target.to.source` dient zum Zurückgeben der Ausgabe von Befehlen und aller Bedienernachrichten, die für den Quellenwarteschlangenmanager generiert werden. Außerdem müssen Sie für jeden Kanal eine Übertragungswarteschlange definieren. Bei dieser Warteschlange handelt es sich um eine lokale Warteschlange, die den Namen des empfangenden Warteschlangenmanagers erhält. Der XMITQ-Name muss mit dem Namen des fernen Warteschlangenmanagers übereinstimmen, damit die ferne Verwaltung funktioniert, es sei denn, Sie verwenden einen WS-Manager-Aliasnamen.Abbildung 18 auf Seite 112 In ist diese Konfiguration zusammengefasst.

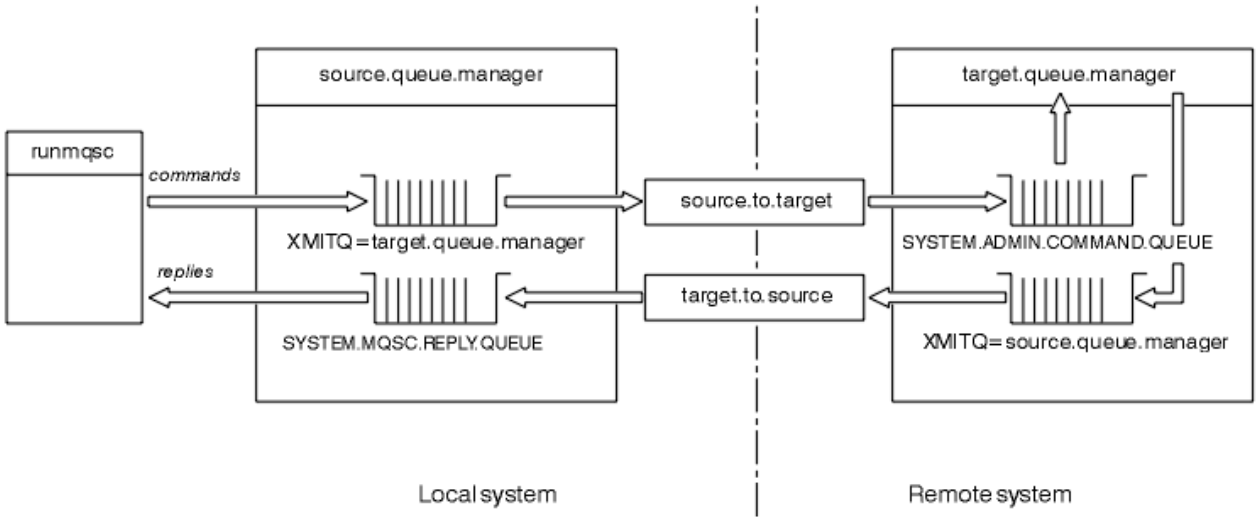


Abbildung 18. Kanäle und Warteschlangen für die Fernverwaltung konfigurieren

Weitere Informationen zum Einrichten von Kanälen finden Sie im Abschnitt Anwendungen über verteilte Steuerung von Warteschlangen verbinden .

Kanäle, Empfangsprogramme und Übertragungswarteschlangen definieren

Geben Sie auf dem Quellenwarteschlangenmanager (`source.queue.manager`) die folgenden MQSC-Befehle aus, um die Kanäle, den Listener und die Übertragungswarteschlange zu definieren:

1. Definieren Sie den Senderkanal auf dem Quellen-WS-Manager:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. Definieren Sie den Empfängerkanal auf dem Quellenwarteschlangenmanager:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Definieren Sie den Listener auf dem Quellen-WS-Manager:

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. Definieren Sie die Übertragungswarteschlange auf dem Quellen-WS-Manager:

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

Geben Sie die folgenden Befehle auf dem Zielwarteschlangenmanager (`target.queue.manager`) aus, um die Kanäle, den Listener und die Übertragungswarteschlange zu erstellen:

1. Definieren Sie den Senderkanal auf dem Ziel-WS-Manager:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. Definieren Sie den Empfängerkanal auf dem Ziel-WS-Manager:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Definieren Sie den Listener auf dem Ziel-WS-Manager:

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. Definieren Sie die Übertragungswarteschlange auf dem Ziel-WS-Manager:

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

Anmerkung: Die für das Attribut CONNAME in den Senderkanaldefinitionen angegebenen TCP/IP-Verbindungsnamen sind nur zur Veranschaulichung bestimmt. Dies ist der Netzname der Maschine am *anderen* Ende der Verbindung. Verwenden Sie die Werte, die für Ihr Netz geeignet sind.

Empfangsprogramme und Kanäle starten

Verwendung von MQSC-Befehlen zum Starten von Empfangsprogrammen und Kanälen.

Starten Sie beide Empfangsprogramme mit den folgenden MQSC-Befehlen:

1. Starten Sie den Listener auf dem Quellen-WS-Manager, `source.queue.manager`, indem Sie den folgenden MQSC-Befehl ausgeben:

```
START LISTENER ('source.queue.manager')
```

2. Starten Sie den Listener auf dem Zielwarteschlangenmanager, `target.queue.manager`, indem Sie den folgenden MQSC-Befehl ausgeben:

```
START LISTENER ('target.queue.manager')
```

Starten Sie beide Senderkanäle, indem Sie die folgenden MQSC-Befehle verwenden:

1. Starten Sie den Senderkanal auf dem Quellenwarteschlangenmanager, `source.queue.manager`, indem Sie den folgenden MQSC-Befehl ausgeben:

```
START CHANNEL ('source.to.target')
```

2. Starten Sie den Senderkanal auf dem Zielwarteschlangenmanager, `target.queue.manager`, indem Sie den folgenden MQSC-Befehl ausgeben:

```
START CHANNEL ('target.to.source')
```

Automatische Definition von Kanälen
Sie aktivieren die automatische Definition von Empfänger-und Serververbindungsdefinitionen, indem Sie das WS-Manager-Objekt mit dem WebSphere MQ-Scriptbefehl ALTER QMGR (oder mit dem PCF-Befehl Change Queue Manager) aktualisieren.

Wenn WebSphere MQ eine eingehende Verbindungsanforderung empfängt und keinen geeigneten Empfänger-bzw. Serververbindungskanal findet, wird automatisch ein Kanal erstellt. Automatische Definitionen basieren auf zwei Standarddefinitionen, die mit WebSphere MQbereitgestellt werden: SYSTEM.AUTO.RECEIVER und SYSTEM.AUTO.SVRCONN

Weitere Informationen zum automatischen Erstellen von Kanaldefinitionen finden Sie unter Kanäle vorbereiten . Informationen zur automatischen Definition von Kanälen für Cluster finden Sie unter Automatische Definition von Clusterkanälen.

Den Befehlsserver für die Fernverwaltung verwalten

Vorgehensweise zum Starten, Stoppen und Anzeigen des Status des Befehlsservers. Ein Befehlsserver ist für die gesamte Verwaltung mit PCF-Befehlen, der MQAI und auch für die Fernverwaltung obligatorisch.

Jedem WS-Manager kann ein Befehlsserver zugeordnet sein. Ein Befehlsserver verarbeitet alle ankommenden Befehle von fernen Warteschlangenmanagern oder PCF-Befehlen aus Anwendungen. Er stellt die Befehle für den Warteschlangenmanager zur Verarbeitung dar und gibt abhängig vom Ursprung des Befehls einen Beendigungscode oder eine Bedienernachricht zurück.

Anmerkung: Stellen Sie für die Fernverwaltung sicher, dass der Zielwarteschlangenmanager aktiv ist. Andernfalls können die Nachrichten, die Befehle enthalten, den WS-Manager nicht verlassen, von dem sie ausgegeben werden. Stattdessen werden diese Nachrichten in die lokale Übertragungswarteschlange eingereiht, die dem fernen Warteschlangenmanager dient. Vermeiden Sie diese Situation.

Es gibt separate Steuerbefehle zum Starten und Stoppen des Befehlsservers. Wenn der Befehlsserver aktiv ist, können Benutzer von WebSphere MQ für Windows oder WebSphere MQ für Linux (x86 -und x86-64 -Plattformen) die in den folgenden Abschnitten beschriebenen Operationen mit IBM WebSphere MQ Explorer ausführen. Weitere Informationen finden Sie unter „Verwaltung mit IBM WebSphere MQ Explorer” auf Seite 58.

Starten des Befehlsservers

Abhängig vom Wert des WS-Managerattributs *SCMDSERV* wird der Befehlsserver entweder automatisch gestartet, wenn der Warteschlangenmanager gestartet wird oder manuell gestartet werden muss. Der Wert des Warteschlangenmanagerattributs kann mit dem MQSC-Befehl ALTER QMGR unter Angabe des Parameters SCMDSERVgeändert werden. Standardmäßig wird der Befehlsserver automatisch gestartet.

Wenn *SCMDSERV* auf MANUAL gesetzt ist, starten Sie den Befehlsserver mit dem folgenden Befehl:

```
strmqcsv saturn.queue.manager
```

Hierbei steht `saturn.queue.manager` für den Warteschlangenmanager, für den der Befehlsserver gestartet wird.

Status des Befehlsservers anzeigen

Stellen Sie für die Fernverwaltung sicher, dass der Befehlsserver auf dem Zielwarteschlangenmanager ausgeführt wird. Wenn der Befehl nicht aktiv ist, können ferne Befehle nicht verarbeitet werden. Alle Nachrichten, die Befehle enthalten, werden in der Befehlswarteschlange des Zielwarteschlangenmanagers in die Warteschlange eingereiht.

Geben Sie den folgenden MQSC-Befehl aus, um den Status des Befehlsservers für einen WS-Manager anzuzeigen:

```
DISPLAY QMSTATUS CMDSERV
```

Befehlsserver stoppen

Verwenden Sie den folgenden Befehl, um den mit dem vorherigen Beispiel gestarteten Befehlsserver zu beenden:

```
endmqcsv saturn.queue.manager
```

Sie können den Befehlsserver auf zwei Arten stoppen:

- Verwenden Sie für einen kontrollierten Stopp den Befehl `endmqcsv` mit dem Flag -c. Dies ist die Standardeinstellung.
- Verwenden Sie für einen sofortigen Stopp den Befehl `endmqcsv` mit dem Flag "-i".

Anmerkung: Wenn Sie einen WS-Manager stoppen, wird auch der ihm zugeordnete Befehlsserver beendet.

MQSC-Befehle auf einem fernen WS-Manager absetzen

Sie können eine bestimmte Form des **`runmqsc`** -Befehls verwenden, um MQSC-Befehle auf einem fernen Warteschlangenmanager auszuführen.

Der Befehlsserver ***muss*** auf dem Ziel-WS-Manager ausgeführt werden, wenn MQSC-Befehle über Remotezugriff verarbeitet werden sollen. (Dies ist auf dem Quellenwarteschlangenmanager nicht erforderlich.) Informationen zum Starten des Befehlsservers auf einem Warteschlangenmanager finden Sie im Abschnitt „Den Befehlsserver für die Fernverwaltung verwalten” auf Seite 114.

Auf dem Quellenwarteschlangenmanager können Sie dann MQSC-Befehle interaktiv im indirekten Modus ausführen, indem Sie Folgendes eingeben:

```
runmqsc -w 30 target.queue.manager
```

Diese Form des Befehls `runmqsc` mit dem Flag -w führt die MQSC-Befehle im indirekten Modus aus, wobei Befehle (in geänderter Form) in die Eingabewarteschlange des Befehlsservers eingereiht und in der Reihenfolge ausgeführt werden.

Wenn Sie einen MQSC-Befehl eingeben, wird er in diesem Fall an den fernen WS-Manager umgeleitet, in diesem Fall `target.queue.manager`. Das Zeitlimit wird auf 30 Sekunden gesetzt. Wenn eine Antwort nicht innerhalb von 30 Sekunden empfangen wird, wird die folgende Nachricht auf dem lokalen (Quellenwarteschlangenmanager) generiert:

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Wenn Sie die Ausgabe von MQSC-Befehlen stoppen, zeigt der lokale WS-Manager alle zeitlich getakten Antworten an, die eingetroffen sind, und löscht alle weiteren Antworten.

Der Quellenwarteschlangenmanager nimmt standardmäßig den lokalen Standardwarteschlangenmanager an. Wenn Sie die Option `-m` *LocalQmgrName* im Befehl **`runmqsc`** angeben, können Sie die Befehle, die ausgegeben werden sollen, über einen beliebigen lokalen Warteschlangenmanager steuern.

Im indirekten Modus können Sie auch eine MQSC-Befehlsdatei auf einem fernen WS-Manager ausführen. Beispiel:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

Hierbei steht `mycomds.in` für eine Datei, die MQSC-Befehle enthält, und `report.out` ist die Berichtsdatei.

Empfohlene Methode zum Fernabsetzen von Befehlen

Wenn Sie Befehle für einen fernen Warteschlangenmanager absetzen, sollten Sie die folgenden Schritte ausführen:

1. Stellen Sie die MQSC-Befehle auf dem fernen System in einer Befehlsdatei zur Ausführung.
2. Überprüfen Sie Ihre MQSC-Befehle lokal, indem Sie das Flag -v im Befehl `runmqsc` angeben.

   Sie können MQSC-Befehle in einem anderen Warteschlangenmanager nicht mit `runmqsc` überprüfen.
3. Überprüfen Sie, ob die Befehlsdatei lokal ohne Fehler ausgeführt wird.
4. Führen Sie die Befehlsdatei auf dem fernen System aus.

Bei Problemen mit MQSC-Befehlen über Remotezugriff

Wenn Sie bei der Ausführung von MQSC-Befehlen über Remotezugriff Schwierigkeiten haben, stellen Sie sicher, dass die folgenden Schritte ausgeführt werden

- Der Befehlsserver wurde auf dem Ziel-WS-Manager gestartet.
- Definiert eine gültige Übertragungswarteschlange.
- Definiert die beiden Enden der Nachrichtenkanäle für beide:
  - Der Kanal, über den die Befehle gesendet werden.
  - Der Kanal, entlang dem die Antworten zurückgegeben werden sollen.
- Geben Sie den korrekten Verbindungsnamen (CONNAME) in der Kanaldefinition an.
- Sie haben die Empfangsprogramme gestartet, bevor Sie die Nachrichtenkanäle gestartet haben.
- Markiert, dass das Unterbrechungsintervall nicht abgelaufen ist, z. B. wenn ein Kanal gestartet wurde, aber dann nach einiger Zeit heruntergefahren wurde. Dies ist besonders wichtig, wenn Sie die Kanäle manuell starten.
- Gesendete Anforderungen von einem Quellenwarteschlangenmanager, die für den Zielwarteschlangenmanager nicht sinnvoll sind (z. B. Anforderungen, die Parameter enthalten, die auf dem fernen Warteschlangenmanager nicht unterstützt werden).

Weitere Informationen hierzu finden Sie im Abschnitt „Probleme mit MQSC-Befehlen beheben” auf Seite 83.

Lokale Definition einer fernen Warteschlange erstellen

Eine lokale Definition einer fernen Warteschlange ist eine Definition in einem lokalen Warteschlangenmanager, die sich auf eine Warteschlange in einem fernen Warteschlangenmanager bezieht.

Sie müssen keine ferne Warteschlange aus einer lokalen Position definieren, aber der Vorteil ist, dass Anwendungen auf die ferne Warteschlange mit ihrem lokal definierten Namen verweisen können, statt einen Namen anzugeben, der durch die ID des Warteschlangenmanagers qualifiziert wird, auf dem sich die ferne Warteschlange befindet.

Informationen zur Funktionsweise lokaler Definitionen von fernen Warteschlangen

Eine Anwendung stellt eine Verbindung zu einem lokalen WS-Manager her und gibt dann einen MQOPEN -Aufruf aus. Im geöffneten Aufruf ist der angegebene Warteschlangenname der Name einer fernen Warte-

schlangendefinition im lokalen Warteschlangenmanager. Die Definition der fernen Warteschlange stellt die Namen der Zielwarteschlange, des Zielwarteschlangenmanagers und optional einer Übertragungswarteschlange bereit. Um eine Nachricht in die ferne Warteschlange einzureihen, gibt die Anwendung einen MQPUT -Aufruf aus, der die vom MQOPEN -Aufruf zurückgegebene Kennung angibt. Der WS-Manager verwendet den Namen der fernen Warteschlange und den Namen des fernen Warteschlangenmanagers in einem Übertragungsheader am Anfang der Nachricht. Diese Informationen werden verwendet, um die Nachricht an ihr korrektes Ziel im Netz weiterzuleiten.

Als Administrator können Sie die Zieladresse der Nachricht steuern, indem Sie die Definition der fernen Warteschlange ändern.

Das folgende Beispiel zeigt, wie eine Anwendung eine Nachricht in eine Warteschlange einreiht, deren Eigner ein ferner WS-Manager ist. Die Anwendung stellt eine Verbindung zu einem WS-Manager her, z. B. `saturn.queue.manager`. Die Zielwarteschlange ist Eigentum eines anderen Warteschlangenmanagers.

Im Aufruf MQOPEN gibt die Anwendung die folgenden Felder an:

Feldwert	**Beschreibung**
ObjectName CYAN.REMOTE.QUEUE	Gibt den lokalen Namen des fernen Warteschlangenobjekts an. Dadurch werden die Zielwarteschlange und der Zielwarteschlangenmanager definiert.
ObjectType (Warteschlange)	Bezeichnet dieses Objekt als Warteschlange.
ObjectQmgrName Leer oder `saturn.queue.manager`	Dieses Feld ist optional. Wenn dieses Feld leer ist, wird der Name des lokalen Warteschlangenmanagers angenommen. (Hierbei handelt es sich um den Warteschlangenmanager, auf dem die Definition der fernen Warteschlange vorhanden ist.)

Danach gibt die Anwendung einen MQPUT -Aufruf aus, um eine Nachricht in diese Warteschlange einzureihen.

Auf dem lokalen WS-Manager können Sie mit den folgenden MQSC-Befehlen eine lokale Definition einer fernen Warteschlange erstellen:

```
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)
```

Dabei gilt:

QREMOTE (CYAN.REMOTE.QUEUE)
Gibt den lokalen Namen des fernen Warteschlangenobjekts an. Dies ist der Name, den Anwendungen, die mit diesem Warteschlangenmanager verbunden sind, im Aufruf MQOPEN angeben müssen, um die Warteschlange AUTOMOBILE.INSURANCE.QUOTE.QUEUE auf dem fernen Warteschlangenmanager `jupiter.queue.manager`.

DESCR ('Queue for auto insurance requests from the branches')
Stellt zusätzlichen Text zur Verfügung, der die Verwendung der Warteschlange beschreibt.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Gibt den Namen der Zielwarteschlange auf dem fernen WS-Manager an. Dies ist die echte Zielwarteschlange für Nachrichten, die von Anwendungen gesendet werden, die den Warteschlangennamen CYAN.REMOTE.QUEUE angeben. Die Warteschlange AUTOMOBILE.INSURANCE.QUOTE.QUEUE muss als lokale Warteschlange auf dem fernen WS-Manager definiert sein.

RQMNAME (jupiter.queue.manager)
Gibt den Namen des fernen Warteschlangenmanagers an, der Eigner der Zielwarteschlange ist AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

XMITQ (INQUOTE.XMIT.QUEUE)
Gibt den Namen der Übertragungswarteschlange an. Dies ist optional. Wenn der Name einer Übertragungswarteschlange nicht angegeben wird, wird eine Warteschlange mit demselben Namen wie der ferne Warteschlangenmanager verwendet.

In beiden Fällen muss die entsprechende Übertragungswarteschlange als lokale Warteschlange mit einem Attribut *Usage* definiert sein, in dem angegeben ist, dass es sich um eine Übertragungswarteschlange (USAGE (XMITQ) in MQSC-Befehlen) handelt.

Eine alternative Möglichkeit, Nachrichten in eine ferne Warteschlange zu stellen

Die Verwendung einer lokalen Definition einer fernen Warteschlange ist nicht die einzige Möglichkeit, Nachrichten in eine ferne Warteschlange zu stellen. Anwendungen können den vollständigen Warteschlangennamen, einschließlich des Namens des fernen Warteschlangenmanagers, als Teil des MQOPEN -Aufrufs angeben. In diesem Fall benötigen Sie keine lokale Definition einer fernen Warteschlange. Dies bedeutet jedoch, dass Anwendungen zur Ausführungszeit entweder den Namen des fernen Warteschlangenmanagers kennen oder Zugriff darauf haben müssen.

Andere Befehle mit fernen Warteschlangen verwenden

Sie können MQSC-Befehle verwenden, um die Attribute eines fernen Warteschlangenobjekts anzuzeigen oder zu ändern, oder Sie können das ferne Warteschlangenobjekt löschen. Beispiel:

- Gehen Sie wie folgt vor, um die Attribute der fernen Warteschlange anzuzeigen

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- Sie können die ferne Warteschlange ändern, um sie zu aktivieren. Dies hat keine Auswirkungen auf die Zielwarteschlange, sondern nur Anwendungen, die diese ferne Warteschlange angeben:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- Zum Löschen dieser fernen Warteschlange. Dies hat keine Auswirkungen auf die Zielwarteschlange, sondern nur ihre lokale Definition:

```
  DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Anmerkung: Wenn Sie eine ferne Warteschlange löschen, löschen Sie nur die lokale Darstellung der fernen Warteschlange. Sie löschen nicht die ferne Warteschlange selbst oder alle Nachrichten darauf.

Übertragungswarteschlange definieren

Eine Übertragungswarteschlange ist eine lokale Warteschlange, die verwendet wird, wenn ein WS-Manager Nachrichten über einen Nachrichtenkanal an einen fernen Warteschlangenmanager weiterleitet.

Der Kanal stellt einen Einweg-Link zum fernen Warteschlangenmanager bereit. Nachrichten werden in der Übertragungswarteschlange in die Warteschlange gestellt, bis der Kanal sie akzeptieren kann. Wenn Sie einen Kanal definieren, müssen Sie einen Namen für die Übertragungswarteschlange an der sendenden Seite des Nachrichtenkanals angeben.

Das MQSC-Befehlsattribut USAGE definiert, ob es sich bei einer Warteschlange um eine Übertragungswarteschlange oder eine normale Warteschlange handelt.

Standardübertragungswarteschlangen

Wenn ein Warteschlangenmanager Nachrichten an einen fernen Warteschlangenmanager sendet, ermittelt er die Übertragungswarteschlange in der folgenden Reihenfolge:

1. Die Übertragungswarteschlange, die im Attribut XMITQ der lokalen Definition einer fernen Warteschlange angegeben ist.
2. Eine Übertragungswarteschlange mit demselben Namen wie der Zielwarteschlangenmanager. (Dieser Wert ist der Standardwert in XMITQ für die lokale Definition einer fernen Warteschlange.)
3. Die Übertragungswarteschlange, die im Attribut DEFXMITQ des lokalen Warteschlangenmanagers angegeben ist.

Der folgende MQSC-Befehl erstellt beispielsweise eine Standardübertragungswarteschlange auf `source.queue.manager` für Nachrichten, die an `target.queue.manager` gesendet werden:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Anwendungen können Nachrichten direkt in eine Übertragungswarteschlange einreihen oder indirekt mithilfe der Definition einer fernen Warteschlange. Weitere Informationen hierzu finden Sie im Abschnitt „Lokale Definition einer fernen Warteschlange erstellen” auf Seite 116.

Ferne Warteschlangendefinitionen als Aliasnamen verwenden

Zusätzlich zum Lokalisieren einer Warteschlange in einem anderen Warteschlangenmanager können Sie auch eine lokale Definition einer fernen Warteschlange für WS-Manager-Aliasnamen und Aliasnamen für Antwortwarteschlangen verwenden. Beide Typen von Aliasnamen werden durch die lokale Definition einer fernen Warteschlange aufgelöst. Sie müssen die entsprechenden Kanäle für die Nachricht einrichten, die an ihrem Ziel ankommen soll.

WS-Manager-Aliasnamen

Ein Aliasname ist der Prozess, durch den der Name des Zielwarteschlangenmanagers, wie in einer Nachricht angegeben, von einem WS-Manager auf dem Nachrichtenweg geändert wird.WS-Manager-Aliasnamen sind wichtig, da Sie sie verwenden können, um das Ziel von Nachrichten in einem Netz von Warteschlangenmanagern zu steuern.

Dies tun Sie, indem Sie die Definition der fernen Warteschlange auf dem Warteschlangenmanager am Steuerungspunkt ändern. Der sendenden Anwendung ist nicht bekannt, dass der angegebene Name des WS-Managers ein Aliasname ist.

Weitere Informationen zu Warteschlangen-Manager-Aliasnamen finden Sie unter Was sind Aliasnamen?.

Antwortwarteschlangenaliasnamen

Optional kann eine Anwendung den Namen einer Warteschlange für Antwortnachrichten angeben, wenn sie eine *Anforderungsnachricht* in eine Warteschlange einreiht.

Wenn die Anwendung, die die Nachricht verarbeitet, den Namen der Warteschlange für Antwortnachrichten extrahiert, weiß sie, wohin die *Antwortnachricht* gesendet werden soll, falls erforderlich.

Ein Aliasname der Empfangswarteschlange für Antworten ist der Prozess, mit dem eine Empfangswarteschlange für Antworten, wie in einer Anforderungsnachricht angegeben, von einem WS-Manager auf dem Nachrichtenweg geändert wird. Der sendenden Anwendung ist nicht bekannt, dass der angegebene Antwortwarteschlangenname ein Aliasname ist.

Mit einem Aliasnamen für die Antwortwarteschlange können Sie den Namen der Empfangswarteschlange für Antworten und optional dessen Warteschlangenmanager ändern. Über diese wiederum können Sie steuern, welche Route für Antwortnachrichten verwendet wird.

Weitere Informationen zu Anforderungsnachrichten, Antwortnachrichten und Antwortwarteschlangen finden Sie unter Nachrichtenarten und Empfangswarteschlange für Antworten und Warteschlangenmanager .

Weitere Informationen zu Aliasnamen für Empfangswarteschlangen für Antworten finden Sie im Abschnitt Aliasnamen und Cluster für Empfangswarteschlangen für Antworten .

Datenkonvertierung zwischen codierten Zeichensätzen

Nachrichtendaten in WebSphere MQ definierten Formaten (auch als *integrierte Formate*bezeichnet) können vom Warteschlangenmanager von einem codierten Zeichensatz in einen anderen konvertiert werden, sofern sich beide Zeichensätze auf eine einzelne Sprache oder eine Gruppe ähnlicher Sprachen beziehen.

Beispielsweise wird die Konvertierung zwischen codierten Zeichensätzen mit Kennungen (CCSIDs) 850 und 500 unterstützt, da beide auf westeuropäisch-sprachige Sprachen anwendbar sind.

Informationen zur Konvertierung von EBCDIC-Zeilenvorschubzeichen (NL) in ASCII finden Sie unter Alle Warteschlangenmanager .

Unterstützte Konvertierungen werden im Abschnitt Datenkonvertierung definiert.

Wenn ein WS-Manager Nachrichten in integrierten Formaten nicht konvertieren kann

Der WS-Manager kann Nachrichten nicht automatisch in integrierte Formate konvertieren, wenn ihre CCSIDs unterschiedliche Landessprachengruppen darstellen. Beispielsweise wird die Konvertierung zwischen CCSID 850 und CCSID 1025 (die ein EBCDIC-codierter Zeichensatz für Sprachen mit kyrillischem Script ist) nicht unterstützt, da viele der Zeichen in einem codierten Zeichensatz nicht in der anderen dargestellt werden können. Wenn Sie über ein Netz von Warteschlangenmanagern in verschiedenen Landessprachen verfügen und die Datenkonvertierung unter einigen der codierten Zeichensätze nicht unterstützt wird, können Sie eine Standardkonvertierung aktivieren. Die Standarddatenkonvertierung wird unter „Standarddatenkonvertierung” auf Seite 120 beschrieben.

Datei 'ccsid.tbl'

Die Datei 'ccsid.tbl' wird für folgende Zwecke verwendet:

- In WebSphere MQ for Fenster werden alle unterstützten codierten Zeichensätze aufgezeichnet.
- Auf AIX- und HP-UX-Plattformen werden die unterstützten codierten Zeichensätze intern vom Betriebssystem gehalten.
- Für alle anderen UNIX and Linux -Plattformen sind die unterstützten codierten Zeichensätze in Konvertierungstabellen enthalten, die von WebSphere MQbereitgestellt werden.
- Sie gibt alle zusätzlichen codierten Zeichensätze an. Bearbeiten Sie die Datei 'ccsid.tbl' (in der Datei selbst ist eine Anleitung enthalten), um zusätzliche Zeichensätze anzugeben.
- Sie gibt eine Standarddatenkonvertierung an.

Die Daten in der Datei 'ccsid.tbl' können aktualisiert werden. Dies kann zum Beispiel sinnvoll sein, wenn ein späteres Release Ihres Betriebssystems zusätzliche codierte Zeichensätze unterstützt.

In WebSphere MQ für Windowsbefindet sich ccsid.tbl standardmäßig im Verzeichnis `C:\Program Files\IBM\WebSphere MQ\conv\table` .

In WebSphere MQ für UNIX and Linux -Systeme befindet sich ccsid.tbl im Verzeichnis `/var/mqm/conv/table`.

Standarddatenkonvertierung

Wenn Sie Kanäle zwischen zwei Maschinen einrichten, auf denen die Datenkonvertierung normalerweise nicht unterstützt wird, müssen Sie die Standarddatenkonvertierung für die Kanäle aktivieren, damit sie funktionieren.

Die Standarddatenkonvertierung wird aktiviert, indem Sie in der Datei 'ccsid.tbl' eine Standard-EBCDIC-CCSID und eine Standard-ASCII-CCSID angeben. Anweisungen dazu, wie Sie dies tun, sind in der Datei enthalten. Sie müssen dies auf allen Maschinen tun, die über die Kanäle verbunden werden. Starten Sie den WS-Manager erneut, damit die Änderung wirksam wird.

Der Standarddatenkonvertierungsprozess ist wie folgt:

- Wenn die Konvertierung zwischen der Quellen-und der Ziel-CCSIDs nicht unterstützt wird, die CCSIDs der Quellen-und Zielumgebungen jedoch entweder EBCDIC oder beide ASCII sind, werden die Zeichendaten ohne Konvertierung an die Zielanwendung übergeben.
- Wenn eine CCSID einen codierten ASCII-Zeichensatz und die andere einen EBCDIC-codierten Zeichensatz darstellt, konvertiert WebSphere MQ die Daten unter Verwendung der in ccsid.tbldefinierten Standard-CCSIDs für Datenkonvertierung.

Anmerkung: Versuchen Sie, die zu konvererenden Zeichen auf diejenigen zu beschränken, die die gleichen Codewerte in dem codierten Zeichensatz haben, der für die Nachricht und den codierten Standardzeichensatz angegeben ist. Wenn Sie nur den Zeichensatz verwenden, der für WebSphere MQ -Objektnamen gültig ist (wie in Naming IBM WebSphere MQ -Objektedefiniert), erfüllen Sie diese Anforderung im Allgemeinen. Es treten Ausnahmebedingungen mit EBCDIC-CCSIDs 290, 930, 1279 und 5026 auf, die in Japan verwendet werden, wobei die Kleinbuchstaben andere Codes als die in anderen EBCDIC-CCSIDs verwendeten Zeichen haben.

Konvertieren von Nachrichten in benutzerdefinierte Formate

Der WS-Manager kann keine Nachrichten in benutzerdefinierten Formaten von einem codierten Zeichensatz in einen anderen konvertieren. Wenn Sie Daten in ein benutzerdefiniertes Format konvertieren müssen, müssen Sie für jedes dieser Formate einen Datenkonvertierungsexit angeben. Verwenden Sie keine Standard-CCSIDs, um Zeichendaten in benutzerdefinierte Formate zu konvertieren. Weitere Informationen zum Konvertieren von Daten in benutzerdefinierte Formate und zum Schreiben von Datenkonvertierungsexits finden Sie im Abschnitt Datenkonvertierungsexits schreiben .

CCSID des Warteschlangenmanagers ändern

Wenn Sie das Attribut CCSID des Befehls ALTER QMGR verwendet haben, um die CCSID des Warteschlangenmanagers zu ändern, stoppen Sie den Warteschlangenmanager und starten Sie ihn erneut, um sicherzustellen, dass alle aktiven Anwendungen, einschließlich des Befehlsservers und des Kanalprogramms, gestoppt und erneut gestartet werden.

Dies ist erforderlich, da alle Anwendungen, die ausgeführt werden, wenn die CCSID des Warteschlangenmanagers geändert wird, weiterhin die vorhandene CCSID verwenden.

IBM WebSphere MQ Telemetry verwalten

IBM WebSphere MQ Telemetry wird über IBM WebSphere MQ Explorer oder über eine Befehlszeile verwaltet. Im Explorer können Sie Telemetriekanäle konfigurieren, den Telemetrieservice steuern und die MQTT-Clients überwachen, die mit IBM WebSphere MQverbunden sind. Konfigurieren Sie die Sicherheit von IBM WebSphere MQ Telemetry mit JAAS, SSL und dem IBM WebSphere MQ -Objektberechtigungsmanager.

Verwaltung mit IBM WebSphere MQ Explorer

Im Explorer können Sie Telemetriekanäle konfigurieren, den Telemetrieservice steuern und die MQTT-Clients überwachen, die mit IBM WebSphere MQverbunden sind. Konfigurieren Sie die Sicherheit von IBM WebSphere MQ Telemetry mit JAAS, SSL und dem IBM WebSphere MQ -Objektberechtigungsmanager.

Verwaltung über die Befehlszeile

IBM WebSphere MQ Telemetry kann vollständig über die Befehlszeiele mithilfe der IBM WebSphere MQMQSC-Befehle verwaltet werden.

Die IBM WebSphere MQ Telemetry -Dokumentation enthält auch Beispielscripts, die die grundlegende Verwendung der MQ Telemetry Transport v3 -Clientanwendung veranschaulichen.

Lesen und verstehen Sie die Beispiele in IBM WebSphere MQ Telemetry-Beispielprogramme vor der Verwendung im Abschnitt Anwendungen für IBM WebSphere MQ Telemetry entwickeln.

Zugehörige Konzepte

WebSphere MQ Telemetry

„Verteilte Steuerung von Warteschlangen für das Senden von Nachrichten an MQTT-Clients konfigurieren" auf Seite 126
IBM WebSphere MQ -Anwendungen können MQTT v3 -Clientnachrichten senden, indem sie in einer Subskription veröffentlichen, die von einem Client erstellt wurde, oder indem sie eine Nachricht direkt senden. Unabhängig von der verwendeten Methode wird die Nachricht in `SYSTEM.MQTT.TRANSMIT.QUEUE`eingereiht und vom Telemetrieservice (MQXR) an den Client gesendet. Es gibt verschiedene Möglichkeiten, eine Nachricht in `SYSTEM.MQTT.TRANSMIT.QUEUE`zu platzieren.

„MQTT-Clientidentifikation, Autorisierung und Authentifizierung" auf Seite 129

„Authentifizierung des Telemetriekanals über SSL" auf Seite 136

„Datenschutz auf Telemetriekanälen" auf Seite 138

„SSL-Konfiguration der MQTT-Clients und Telemetriekanäle" auf Seite 139

„JAAS-Konfiguration für Telemetriekanal" auf Seite 144
Konfigurieren Sie JAAS für die Authentifizierung des vom Client gesendeten `Benutzernamens` .

„Konzepte des IBM WebSphere MQ Telemetry-Dämons für Geräte" auf Seite 146
Der IBM WebSphere MQ Telemetry-Dämon für Geräte ist eine erweiterte MQTT V3 -Clientanwendung. Damit können Sie Nachrichten von anderen MQTT-Clients speichern und weiterleiten (Store-and-forward-Verfahren). Der Dämon stellt wie ein MQTT-Client eine Verbindung zu IBM WebSphere MQ her, Sie können jedoch auch andere MQTT-Clients mit ihm verbinden.

Zugehörige Tasks

„Warteschlangenmanager für Telemetrie unter Linux und AIX konfigurieren" auf Seite 122
Führen Sie diese Schritte aus, um einen Warteschlangenmanager manuell für die Ausführung von IBM WebSphere MQ Telemetry zu konfigurieren. Sie können eine automatisierte Prozedur ausführen, um eine einfachere Konfiguration mithilfe der IBM WebSphere MQ Telemetry -Unterstützung für IBM WebSphere MQ Explorer einzurichten.

„Warteschlangenmanager für Telemetrie unter Windows konfigurieren" auf Seite 124
Führen Sie diese Schritte aus, um einen Warteschlangenmanager manuell für die Ausführung von IBM WebSphere MQ Telemetry zu konfigurieren. Sie können eine automatisierte Prozedur ausführen, um eine einfachere Konfiguration mithilfe der IBM WebSphere MQ Telemetry -Unterstützung für IBM WebSphere MQ Explorer einzurichten.

Zugehörige Verweise

MQXR-Eigenschaften

Warteschlangenmanager für Telemetrie unter Linux und AIX konfigurieren

Führen Sie diese Schritte aus, um einen Warteschlangenmanager manuell für die Ausführung von IBM WebSphere MQ Telemetry zu konfigurieren. Sie können eine automatisierte Prozedur ausführen, um eine einfachere Konfiguration mithilfe der IBM WebSphere MQ Telemetry -Unterstützung für IBM WebSphere MQ Explorer einzurichten.

Vorbereitende Schritte

1. Informationen zur Installation von IBM WebSphere MQund des Features IBM WebSphere MQ Telemetry finden Sie unter IBM WebSphere MQ Telemetryinstallieren .
2. Erstellen und starten Sie einen WS-Manager. Der WS-Manager wird in dieser Task als *qMgr* bezeichnet.
3. Im Rahmen dieser Task konfigurieren Sie den Telemetrieservice (MQXR). Die MQXR-Eigenschaftseinstellungen werden in einer plattformspezifischen Eigenschaftendatei gespeichert: `mqxr_unix.properties`. Normalerweise müssen Sie die MQXR-Eigenschaftendatei nicht direkt bearbeiten, da fast

alle Einstellungen über MQSC-Verwaltungsbefehle oder MQ Explorer konfiguriert werden können. Wenn Sie die Datei direkt bearbeiten möchten, stoppen Sie den Warteschlangenmanager, bevor Sie Ihre Änderungen vornehmen. Siehe MQXR-Eigenschaften.

Informationen zu diesem Vorgang

Die IBM WebSphere MQ Telemetry -Unterstützung für IBM WebSphere MQ Explorer enthält einen Assistenten und eine Beispielbefehlsprozedur `sampleMQM`. Diese richten eine Erstkonfiguration mit der Gastbenutzer-ID ein (siehe Installation von IBM WebSphere MQ Telemetry mithilfe von IBM WebSphere MQ Explorer und IBM WebSphere MQ Telemetry-Beispielprogrammen überprüfen.

Führen Sie die Schritte in dieser Task aus, um IBM WebSphere MQ Telemetry manuell mit verschiedenen Berechtigungsschemas zu konfigurieren.

Vorgehensweise

1. Öffnen Sie ein Befehlsfenster auf dem Telemetriemusterverzeichnis.

   Das Telemetry-Beispielverzeichnis ist `/opt/mqm/mqxr/samples`.
2. Erstellen Sie die Übertragungswarteschlange für Telemetrie.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc qMgr
   ```

   Wenn der Telemetrieservice (MQXR) zuerst gestartet wird, wird `SYSTEM.MQTT.TRANSMIT.QUEUE` erstellt.

   Sie wird in dieser Task manuell erstellt, da `SYSTEM.MQTT.TRANSMIT.QUEUE` vorhanden sein muss, bevor der Telemetrieservice (MQXR) gestartet wird, um den Zugriff auf sie zu autorisieren.
3. Standardübertragungswarteschlange festlegen

   Wenn der Telemetrieservice (MQXR) zum ersten gestartet wird, ändert er den Warteschlangenmanager nicht, um `SYSTEM.MQTT.TRANSMIT.QUEUE` zur Standardübertragungswarteschlange zu machen.

   Um `SYSTEM.MQTT.TRANSMIT.QUEUE` zu der Standardübertragungswarteschlange zu machen, ändern Sie die Eigenschaft der Standardübertragungswarteschlange. Ändern Sie die Eigenschaft mit IBM WebSphere MQ Explorer oder mit dem Befehl im folgenden Beispiel:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   Das Ändern der Standardübertragungswarteschlange kann die vorhandene Konfiguration beeinträchtigen. Die Änderung der Standardübertragungswarteschlange in `SYSTEM.MQTT.TRANSMIT.QUEUE` hat den Grund, dass das Senden von Nachrichten direkt an MQTT-Clients vereinfacht wird. Ohne Änderung der Standardübertragungswarteschlange müssen Sie für jeden Client, der IBM WebSphere MQ-Nachrichten empfängt, eine Definition einer fernen Warteschlange hinzufügen (siehe „Eine Nachricht direkt an einen Client senden” auf Seite 127).
4. Führen Sie eine Prozedur in „MQTT-Clients für den Zugriff auf WebSphere MQ -Objekte berechtigen” auf Seite 130 aus, um eine oder mehrere Benutzer-IDs zu erstellen. Die Benutzer-IDs verfügen über die Berechtigung zum Veröffentlichen, Subskribieren und Senden von Veröffentlichungen an MQTT-Clients.
5. Installieren Sie den Telemetrieservice (MQXR).

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   Siehe auch den Beispielcode in Abbildung 19 auf Seite 124.
6. Starten Sie den Service.

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   Der Telemetrieservice (MQXR) wird automatisch gestartet, wenn der WS-Manager gestartet wird.

Diese Task wird manuell in dieser Task gestartet, da der Warteschlangenmanager bereits ausgeführt wird.

7. Konfigurieren Sie mithilfe von IBM WebSphere MQ ExplorerTelemetriekanäle für das Akzeptieren von Verbindungen von MQTT-Clients.

   Die Telemetriekanäle müssen so konfiguriert sein, dass ihre Identitäten eine der in Schritt 4 definierten Benutzer-IDs sind.

   Siehe auch DEFINE CHANNEL (MQTT).
8. Überprüfen Sie die Konfiguration, indem Sie den Beispielclient ausführen.

   Damit der Beispielclient mit Ihrem Telemetriekanal arbeiten kann, muss der Kanal den Client für die Veröffentlichung, Subskription und Empfang von Veröffentlichungen berechtigen. Der Beispielclient stellt die Verbindung zum Telemetriekanal an Port 1883 standardmäßig her. Siehe auch IBM WebSphere MQ Telemetry-Beispielprogramme.

Beispiel

Abbildung 19 auf Seite 124 zeigt den Befehl **runmqsc** zum manuellen Erstellen der SYSTEM.MQXR.SERVICE unter 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')
```

Abbildung 19. installMQXRService_unix.mqsc

Warteschlangenmanager für Telemetrie unter Windows konfigurieren

Führen Sie diese Schritte aus, um einen Warteschlangenmanager manuell für die Ausführung von IBM WebSphere MQ Telemetry zu konfigurieren. Sie können eine automatisierte Prozedur ausführen, um eine einfachere Konfiguration mithilfe der IBM WebSphere MQ Telemetry -Unterstützung für IBM WebSphere MQ Explorer einzurichten.

Vorbereitende Schritte

1. Informationen zur Installation von IBM WebSphere MQund des Features IBM WebSphere MQ Telemetry finden Sie unter IBM WebSphere MQ Telemetryinstallieren .
2. Erstellen und starten Sie einen WS-Manager. Der WS-Manager wird in dieser Task als *qMgr* bezeichnet.
3. Im Rahmen dieser Task konfigurieren Sie den Telemetrieservice (MQXR). Die MQXR-Eigenschaftseinstellungen werden in einer plattformspezifischen Eigenschaftendatei gespeichert: mqxr_win.properties. Normalerweise müssen Sie die MQXR-Eigenschaftendatei nicht direkt bearbeiten, da fast alle Einstellungen über MQSC-Verwaltungsbefehle oder MQ Explorer konfiguriert werden können. Wenn Sie die Datei direkt bearbeiten möchten, stoppen Sie den Warteschlangenmanager, bevor Sie Ihre Änderungen vornehmen. Siehe MQXR-Eigenschaften.

Informationen zu diesem Vorgang

Die IBM WebSphere MQ Telemetry -Unterstützung für IBM WebSphere MQ Explorer enthält einen Assistenten und eine Beispielbefehlsprozedur sampleMQM. Diese richten eine Erstkonfiguration mit der Gastbenutzer-ID ein (siehe Installation von IBM WebSphere MQ Telemetry mithilfe von IBM WebSphere MQ Explorer und IBM WebSphere MQ Telemetry-Beispielprogrammen überprüfen.

Führen Sie die Schritte in dieser Task aus, um IBM WebSphere MQ Telemetry manuell mit verschiedenen Berechtigungsschemas zu konfigurieren.

Vorgehensweise

1. Öffnen Sie ein Befehlsfenster auf dem Telemetriemusterverzeichnis.

   Das Verzeichnis "telemetry samples" ist *WMQ program installation directory*\mqxr\samples.
2. Erstellen Sie die Übertragungswarteschlange für Telemetrie.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   Wenn der Telemetrieservice (MQXR) zuerst gestartet wird, wird SYSTEM.MQTT.TRANSMIT.QUEUE erstellt.

   Sie wird in dieser Task manuell erstellt, da SYSTEM.MQTT.TRANSMIT.QUEUE vorhanden sein muss, bevor der Telemetrieservice (MQXR) gestartet wird, um den Zugriff auf sie zu autorisieren.
3. Standardübertragungswarteschlange festlegen

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   Abbildung 20. Standardübertragungswarteschlange festlegen

   Wenn der Telemetrieservice (MQXR) zum ersten gestartet wird, ändert er den Warteschlangenmanager nicht, um SYSTEM.MQTT.TRANSMIT.QUEUE zur Standardübertragungswarteschlange zu machen.

   Um SYSTEM.MQTT.TRANSMIT.QUEUE zu der Standardübertragungswarteschlange zu machen, ändern Sie die Eigenschaft der Standardübertragungswarteschlange. Ändern Sie die Eigenschaft mit dem IBM WebSphere MQ Explorer oder mit dem Befehl in Abbildung 20 auf Seite 125.

   Das Ändern der Standardübertragungswarteschlange kann die vorhandene Konfiguration beeinträchtigen. Die Änderung der Standardübertragungswarteschlange in SYSTEM.MQTT.TRANSMIT.QUEUE hat den Grund, dass das Senden von Nachrichten direkt an MQTT-Clients vereinfacht wird. Ohne Änderung der Standardübertragungswarteschlange müssen Sie für jeden Client, der IBM WebSphere MQ-Nachrichten empfängt, eine Definition einer fernen Warteschlange hinzufügen (siehe „Eine Nachricht direkt an einen Client senden” auf Seite 127).
4. Führen Sie eine Prozedur in „MQTT-Clients für den Zugriff auf WebSphere MQ -Objekte berechtigen” auf Seite 130 aus, um eine oder mehrere Benutzer-IDs zu erstellen. Die Benutzer-IDs verfügen über die Berechtigung zum Veröffentlichen, Subskribieren und Senden von Veröffentlichungen an MQTT-Clients.
5. Installieren Sie den Telemetrieservice (MQXR).

   ```
   type
   installMQXRService_win.mqsc | runmqsc qMgr
   ```
6. Starten Sie den Service.

   ```
   echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
   ```

   Der Telemetrieservice (MQXR) wird automatisch gestartet, wenn der WS-Manager gestartet wird.

   Diese Task wird manuell in dieser Task gestartet, da der Warteschlangenmanager bereits ausgeführt wird.
7. Konfigurieren Sie mithilfe von IBM WebSphere MQ ExplorerTelemetriekanäle für das Akzeptieren von Verbindungen von MQTT-Clients.

   Die Telemetriekanäle müssen so konfiguriert sein, dass ihre Identitäten eine der in Schritt 4 definierten Benutzer-IDs sind.

   Siehe auch DEFINE CHANNEL (MQTT).
8. Überprüfen Sie die Konfiguration, indem Sie den Beispielclient ausführen.

   Damit der Beispielclient mit Ihrem Telemetriekanal arbeiten kann, muss der Kanal den Client für die Veröffentlichung, Subskription und Empfang von Veröffentlichungen berechtigen. Der Beispielclient

stellt die Verbindung zum Telemetriekanal an Port 1883 standardmäßig her. Siehe auch IBM WebSphere MQ Telemetry-Beispielprogramme.

SYSTEM.MQXR.SERVICE manuell erstellen

Abbildung 21 auf Seite 126 zeigt den Befehl **runmqsc** zum manuellen Erstellen von SYSTEM.MQXR.SERVICE unter 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')
```

Abbildung 21. installMQXRService_win.mqsc

Verteilte Steuerung von Warteschlangen für das Senden von Nachrichten an MQTT-Clients konfigurieren

IBM WebSphere MQ -Anwendungen können MQTT v3 -Clientnachrichten senden, indem sie in einer Subskription veröffentlichen, die von einem Client erstellt wurde, oder indem sie eine Nachricht direkt senden. Unabhängig von der verwendeten Methode wird die Nachricht in SYSTEM.MQTT.TRANSMIT.QUEUEeingereiht und vom Telemetrieservice (MQXR) an den Client gesendet. Es gibt verschiedene Möglichkeiten, eine Nachricht in SYSTEM.MQTT.TRANSMIT.QUEUEzu platzieren.

Nachricht als Antwort auf eine MQTT-Clientsubskription veröffentlichen

Der Telemetrieservice (MQXR) erstellt eine Subskription für den MQTT-Client. Der Client ist das Ziel für alle Veröffentlichungen, die mit der Subskription übereinstimmen, die vom Client gesendet wurde. Die Telemetrie-Services leiten übereinstimmende Veröffentlichungen wieder an den Client weiter.

Ein MQTT-Client ist mit WebSphere MQ als Warteschlangenmanager verbunden, dessen Name des Warteschlangenmanagers auf seine ClientIdentifiergesetzt ist. Die Zieladresse für Veröffentlichungen, die an den Client gesendet werden sollen, ist eine Übertragungswarteschlange, SYSTEM.MQTT.TRANSMIT.QUEUE. Der Telemetrieservice leitet Nachrichten in SYSTEM.MQTT.TRANSMIT.QUEUE an MQTT-Clients weiter, wobei der Name des Zielwarteschlangenmanagers als Schlüssel für einen bestimmten Client verwendet wird.

Der Telemetrieservice (MQXR) öffnet die Übertragungswarteschlange unter Verwendung von ClientIdentifier als Name des Warteschlangenmanagers. Der Telemetrieservice (MQXR) übergibt die Objektkennung der Warteschlange an den Aufruf MQSUB , um Veröffentlichungen weiterzuleiten, die der Clientsubskription entsprechen. Bei der Auflösung des Objektnamens wird ClientIdentifier als Name des fernen Warteschlangenmanagers erstellt und die Übertragungswarteschlange muss in SYSTEM.MQTT.TRANSMIT.QUEUEaufgelöst werden. Bei Verwendung der WebSphere MQ -Standardobjektnamensauflösung wird *ClientIdentifier* wie folgt aufgelöst (siehe Tabelle 6 auf Seite 127).

1. *ClientIdentifier* stimmt mit nichts überein.

   ClientIdentifier ist ein Name des fernen Warteschlangenmanagers. Er stimmt nicht mit dem Namen des lokalen Warteschlangenmanagers, einem WS-Manager-Aliasnamen oder einem Übertragungswarteschlangennamen überein.

   Der Warteschlangenname ist nicht definiert. Derzeit setzt der Telemetrieservice (MQXR) SYSTEM.MQTT.PUBLICATION.QUEUE als Namen der Warteschlange. Ein MQTT v3 -Client unterstützt keine Warteschlangen, sodass der aufgelöste Warteschlangenname vom Client ignoriert wird.

   Die Eigenschaft Standardübertragungswarteschlangedes lokalen Warteschlangenmanagers muss auf SYSTEM.MQTT.TRANSMIT.QUEUEgesetzt werden, damit die Veröffentlichung in SYSTEM.MQTT.TRANSMIT.QUEUE eingereiht und an den Client gesendet wird.

2. *ClientIdentifier* stimmt mit einem WS-Manager-Aliasnamen mit dem Namen *ClientIdentifier* überein

   ClientIdentifier ist ein Name des fernen Warteschlangenmanagers. Er stimmt mit dem Namen eines WS-Manager-Aliasnamens überein.

   Der Aliasname des WS-Managers muss mit *ClientIdentifier* als Name des fernen Warteschlangenmanagers definiert werden.

   Wenn Sie den Namen der Übertragungswarteschlange in der WS-Manager-Aliasdefinition festlegen, ist es nicht erforderlich, dass die Standardübertragung auf SYSTEM.MQTT.TRANSMIT.QUEUE gesetzt wird.

Tabelle 6. Namensauflösung eines MQTT-WS-Manager-Aliasnamens

	Eingabe		**Ausgabe**		
ClientIdentifier	**Name des Warteschlangenmanagers**	**Warteschlangenname**	**Name des Warteschlangenmanagers**	**Warteschlangenname**	**Übertragungswarteschlange**
Stimmt mit nichts überein	*ClientIdentifier*	*undefined*	*ClientIdentifier*	*undefined*	Standardübertragungswarteschlange. SYSTEM.MQTT.TRANSMIT.QUEUE
Übereinstimmung mit einem WS-Manager-Aliasnamen mit dem Namen *ClientIdentifier*	*ClientIdentifier*	*undefined*	*ClientIdentifier*	*undefined*	SYSTEM.MQTT.TRANSMIT.QUEUE

Weitere Informationen zur Namensauflösung finden Sie in Namensauflösung .

Jedes WebSphere MQ -Programm kann zu demselben Thema veröffentlichen. Die Veröffentlichung wird an ihre Subskribenten gesendet, einschließlich MQTT v3 -Clients, die über eine Subskription für das Topic verfügen.

Wenn in einem Cluster ein Verwaltungsthema mit dem Attribut CLUSTER(*clusterName*) erstellt wird, kann jede beliebige Anwendung im Cluster auf dem Client veröffentlicht werden, z. B.:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

Abbildung 22. Clusterthema unter Windows definieren

Anmerkung: Geben Sie SYSTEM.MQTT.TRANSMIT.QUEUE kein Clusterattribut an.

MQTT-Clientsubskribenten und Publisher können Verbindungen zu verschiedenen Warteschlangenmanagern herstellen. Subskribenten und Publisher können Teil desselben Clusters sein oder durch eine Publish/Subscribe-Hierarchie verbunden sein. Die Veröffentlichung wird über WebSphere MQvom Publisher an den Subskribenten zugestellt.

Eine Nachricht direkt an einen Client senden

Als Alternative zu einem Client, der eine Subskription erstellt und eine Veröffentlichung empfängt, die dem Subskriptionsthema entspricht, senden Sie eine Nachricht direkt an einen MQTT v3 -Client. MQTT-

V3 -Clientanwendungen können keine Nachrichten direkt senden, aber andere Anwendungen, wie z. B. WebSphere MQ -Anwendungen, können dies.

Die Anwendung WebSphere MQ muss die `ClientIdentifier` des MQTT- v3 -Clients kennen. Da MQTT v3 -Clients keine Warteschlangen haben, wird der Name der Zielwarteschlange als Topicname an die MQTT v3 -Anwendungsclient `messageArrived` -Methode übergeben. Erstellen Sie zum Beispiel in einem MQI-Programm einen Objektdeskriptor mit dem Client als `ObjectQmgrName` :

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

Abbildung 23. MQI-Objektdeskriptor zum Senden einer Nachricht an ein MQTT- v3 -Clientziel

Wenn die Anwendung mit JMS geschrieben wird, erstellen Sie ein Punkt-zu-Punkt-Ziel. Beispiel:

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

Abbildung 24. JMS-Ziel zum Senden einer Nachricht an einen MQTT- v3 -Client

Verwenden Sie zum Senden einer nicht angeforderten Nachricht an einen MQTT-Client eine Definition einer fernen Warteschlange. Der Name des fernen Warteschlangenmanagers muss in die Client-ID (`ClientIdentifier`) des Clients aufgelöst werden. Die Übertragungswarteschlange muss in SYSTEM.MQTT.TRANSMIT.QUEUEaufgelöst werden (siehe Tabelle 7 auf Seite 128). Der Name der fernen Warteschlange kann alles sein. Der Client empfängt ihn als Themenzeichenfolge.

Tabelle 7. Namensauflösung einer Definition einer fernen Warteschlange des MQTT-Clients

Eingabe		**Ausgabe**		
Warteschlangenname	**Name des Warteschlangenmanagers**	**Warteschlangenname**	**Name des Warteschlangenmanagers**	**Übertragungswarteschlange**
Name der Definition der fernen Warteschlange	Leerer oder lokaler WS-Managername	Name der fernen Warteschlange, die als Themenzeichenfolge verwendet wird	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

Wenn der Client verbunden ist, wird die Nachricht direkt an den MQTT-Client gesendet, der die Methode `messageArrived` aufruft (siehe `messageArrived` -Methode).

Wenn der Client die Verbindung zu einer persistenten Sitzung getrennt hat, wird die Nachricht in SYSTEM.MQTT.TRANSMIT.QUEUEgespeichert; siehe Statusunabhängige und statusabhängige MQTT-Sitzungen . Er wird an den Client weitergeleitet, wenn der Client die Verbindung erneut zu der Sitzung wiederherstellt.

Wenn Sie eine nicht persistente Nachricht senden, wird sie mit der Servicequalität "at most once" (Höchstens einmal) an den Client gesendet (QoS=0). Wenn Sie eine persistente Nachricht direkt an einen Client senden, wird diese standardmäßig mit der Servicequalität "exactly once" (Genau einmal) gesendet (QoS=2). Da der Client möglicherweise keinen Persistenzmechanismus hat, kann der Client die Servicequalität verringern, die er für Nachrichten akzeptiert, die direkt gesendet werden. Um die Servicequalität für Nachrichten, die direkt an einen Client gesendet werden, zu verringern, nehmen Sie eine Subskription für das Thema DEFAULT.QoS vor. Geben Sie die maximale Servicequalität an, die der Client unterstützen kann.

MQTT-Clientidentifikation, Autorisierung und Authentifizierung

Der Telemetrieservice (MQXR) veröffentlicht oder subskribiert unter Verwendung von MQTT-Kanälen WebSphere MQ-Themen für MQTT-Clients. Der WebSphere MQ-Administrator konfiguriert die MQTT-Kanalidentität, die für die WebSphere MQ-Berechtigung verwendet wird. Der Administrator kann eine gemeinsame Identität für den Kanal definieren oder den Benutzernamen oder den ClientIdentifier eines Clients verwenden, der mit dem Kanal verbunden ist.

Der Telemetrieservice (MQXR) kann den Client über den Benutzernamen authentifizieren, der vom Client bereitgestellt wird, oder indem er ein Clientzertifikat verwendet. Der Benutzername wird mit einem Kennwort authentifiziert, das vom Client bereitgestellt wird.

Zusammenfassen: Die Client-ID ist die Auswahl der Cliendentität. Je nach Kontext wird der Client durch die Client-ID , Benutzername , eine vom Administrator erstellte gemeinsame Cliendentität oder ein Clientzertifikat identifiziert. Die für die Authentizitätsprüfung verwendete Client-ID muss nicht mit der ID identisch sein, die für die Autorisierung verwendet wird.

MQTT-Clientprogramme legen die Angaben für den Benutzernamen und das Kennwort fest, die an den Server unter Verwendung eines MQTT-Kanals gesendet werden. Sie können auch die SSL-Eigenschaften festlegen, die zur Verschlüsselung und Authentifizierung der Verbindung erforderlich sind. Der Administrator entscheidet, ob und wie der MQTT-Kanal authentifiziert wird.

Zur Autorisierung eines MQTT-Clients für den Zugriff auf WebSphere MQ-Objekte müssen Sie die Client-ID oder den Benutzernamen des Clients oder eine allgemeine Client-ID autorisieren. Wenn Sie einem Client die Verbindung mit WebSphere MQ ermöglichen möchten, authentifizieren Sie den Benutzernamen oder verwenden Sie ein Clientzertifikat. Konfigurieren Sie JAAS, um den Benutzernamen zu authentifizieren, und konfigurieren Sie SSL, um ein Clientzertifikat zu authentifizieren.

Wenn Sie ein Kennwort beim Client festlegen, verschlüsseln Sie die Verbindung entweder über ein virtuelles privates Netz (VPN) oder konfigurieren Sie den MQTT-Kanal für die Verwendung von SSL, um das Kennwort nicht öffentlich zu machen.

Es ist schwierig, Clientzertifikate zu verwalten. Aus diesem Grund wird die Kennwortauthentifizierung häufig zur Authentifizierung von Clients verwendet, wenn die Risiken, die mit der Kennwortauthentifizierung verknüpft sind, akzeptabel sind.

Wenn es eine sichere Möglichkeit gibt, das Clientzertifikat zu verwalten und zu speichern, ist es möglich, sich auf die Zertifikatsauthentifizierung zu verlassen. Es ist jedoch selten der Fall, dass Zertifikate sicher in den Typen von Umgebungen verwaltet werden können, in denen Telemetrie verwendet wird. Stattdessen wird die Authentifizierung von Einheiten, die Clientzertifikate verwenden, durch die Authentifizierung von Clientkennwörtern auf dem Server ergänzt. Aufgrund der zusätzlichen Komplexität ist die Verwendung von Clientzertifikaten auf hochsensible Anwendungen beschränkt. Die Verwendung von zwei Formen der Authentifizierung wird als Zwei-Faktor-Authentifizierung bezeichnet. Sie müssen einen der Faktoren kennen, z. B. ein Kennwort, und die anderen Faktoren, wie z. B. ein Zertifikat, haben.

Bei einer hochempfindlichen Anwendung, wie z. B. einer Chip-und-Pin-Vorrichtung, wird die Vorrichtung während der Fertigung gesperrt, um Manipulationen an der internen Hardware und Software zu verhindern. Ein anerkantes, zeitbegrenztes Clientzertifikat wird in die Einheit kopiert. Die Einheit wird an der Position implementiert, an der sie verwendet werden soll. Die weitere Authentifizierung wird jedes Mal durchgeführt, wenn die Einheit verwendet wird, entweder mit einem Kennwort oder einem anderen Zertifikat von einer Smart-Card.

MQTT-Clientidentität und Autorisierung

Verwenden Sie die Client-ID, den Benutzernamen oder eine allgemeine Clientidentität für die Autorisierung des Zugriffs auf WebSphere MQ-Objekte.

Der WebSphere MQ-Administrator hat drei Möglichkeiten zur Auswahl der Identität des MQTT-Kanals. Er trifft diese Auswahl beim Definieren oder Ändern des MQTT-Kanals, der vom Client verwendet wird. Mit der Identität wird der Zugriff auf WebSphere MQ-Themen autorisiert. Es sind folgende Auswahlmöglichkeiten verfügbar:

1. Die Client-ID.

2. Eine Identität, die der Administrator für den Kanal zur Verfügung stellt.
3. Der vom MQTT-Client übergebene `Benutzername`.

Der Benutzername (`Username`) ist ein Attribut der Klasse 'MqttConnectOptions'. Es muss festgelegt werden, bevor der Client eine Verbindung zum Service herstellt. Sein Standardwert lautet null.

Wählen Sie mit dem WebSphere MQ-Befehl **setmqaut** aus, für welche Objekte und welche Aktionen die Identität berechtigt sein soll, die dem MQTT-Kanal zugeordnet ist. Geben Sie beispielsweise Folgendes ein, um die Kanalidentität `MQTTClient`, die vom Administrator des Warteschlangenmanagers `QM1` bereitgestellt wird, zu autorisieren:

```
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
```

MQTT-Clients für den Zugriff auf WebSphere MQ -Objekte berechtigen

Führen Sie die folgenden Schritte aus, um MQTT-Clients für die Veröffentlichung und Subskription von WebSphere MQ -Objekten zu berechtigen. Die Schritte folgen vier alternativen Zugriffssteuerungsmustern.

Vorbereitende Schritte

MQTT-Clients sind berechtigt, auf Objekte in WebSphere MQ zuzugreifen, indem ihnen eine Identität zugeordnet wird, wenn sie eine Verbindung zu einem Telemetriekanal herstellen. Der WebSphere MQ -Administrator konfiguriert den Telemetriekanal mithilfe von WebSphere MQ Explorer, um einem Client eine von drei Identitätsarten zuzuweisen:

1. `ClientIdentifier`
2. `Benutzername`
3. Ein Name, den der Administrator dem Kanal zuweist.

Unabhängig davon, welcher Typ verwendet wird, muss die Identität für WebSphere MQ als Principal vom installierten Berechtigungsservice definiert werden. Der Standardberechtigungsservice unter Windows oder Linux wird als Object Authority Manager (OAM) bezeichnet. Wenn Sie den OAM verwenden, muss die Identität als Benutzer-ID definiert sein.

Verwenden Sie die Identität, um einem Client oder einer Gruppe von Clients die Berechtigung zum Veröffentlichen oder Subskribieren von Themen zu erteilen, die in WebSphere MQdefiniert sind. Wenn ein MQTT-Client ein Thema subskribiert hat, verwenden Sie die Identität, um ihm die Berechtigung zum Empfangen der resultierenden Veröffentlichungen zu erteilen.

Es ist schwierig, ein System mit Zehntausenden MQTT-Clients zu verwalten, die jeweils individuelle Zugriffsberechtigungen erfordern. Eine Lösung besteht darin, gemeinsame Identitäten zu definieren und einzelne MQTT-Clients einer der gemeinsamen Identitäten zuzuordnen. Definieren Sie so viele gemeinsame Identitäten wie erforderlich, um unterschiedliche Kombinationen von Berechtigungen zu definieren. Eine andere Lösung besteht darin, einen eigenen Berechtigungsservice zu schreiben, der mit Tausenden von Benutzern einfacher umgehen kann als das Betriebssystem.

Sie können MQTT-Clients mit dem OAM auf zwei Arten zu gemeinsamen Identitäten kombinieren:

1. Definieren Sie mehrere Telemetriekanäle mit jeweils einer anderen Benutzer-ID, die der Administrator mithilfe von WebSphere MQ Explorer zuordnet. Clients, die unter Verwendung unterschiedlicher TCP/IP-Portnummern eine Verbindung herstellen, sind verschiedenen Telemetriekanälen zugeordnet und weisen unterschiedliche Identitäten auf.
2. Definieren Sie einen einzelnen Telemetriekanal, aber lassen Sie jeden Client einen `Benutzernamen` aus einer kleinen Gruppe von Benutzer-IDs auswählen. Der Administrator konfiguriert den Telemetriekanal, um den Client `Username` als Identität auszuwählen.

In dieser Task wird die Identität des Telemetriekanals *`mqttUser`*, unabhängig davon, wie er festgelegt ist, aufgerufen. Wenn Sammlungen von Clients unterschiedliche Identitäten verwenden, verwenden Sie mehrere *`mqttUsers`*, eine für jede Sammlung von Clients. Da die Task den OAM verwendet, muss jede *`mqttUser`* eine Benutzer-ID sein.

Informationen zu diesem Vorgang

In dieser Task haben Sie eine Auswahl von vier Zugriffssteuerungsmustern, die Sie an bestimmte Anforderungen anpassen können. Die Muster unterscheiden sich in ihrer Granularität der Zugriffssteuerung.

- „Keine Zugriffssteuerung” auf Seite 131
- „Coarse-Grained-Zugriffssteuerung” auf Seite 131
- „Mittlere Zugriffssteuerung” auf Seite 131
- „Feinkörnige Zugriffssteuerung” auf Seite 131

Das Ergebnis der Modelle besteht darin, *mqttUsers* Gruppen von Berechtigungen zum Veröffentlichen und Subskribieren in WebSphere MQzuzuordnen und Veröffentlichungen von WebSphere MQzu empfangen.

Keine Zugriffssteuerung
MQTT-Clients erhalten die WebSphere MQ -Administratorberechtigung und können jede Aktion für jedes Objekt ausführen.

Vorgehensweise

1. Erstellen Sie die Benutzer-ID *mqttUser* als Identität aller MQTT-Clients.
2. Fügen Sie *mqttUser* zur Gruppe `mqm` hinzu (siehe Benutzer zu einer Gruppe unter Windows hinzufügenoder Benutzer zu einer Gruppe unter Linuxhinzufügen).

Coarse-Grained-Zugriffssteuerung
MQTT-Clients sind zum Veröffentlichen und Subskribieren sowie zum Senden von Nachrichten an MQTT-Clients berechtigt. Sie verfügen nicht über die Berechtigung zum Ausführen anderer Aktionen oder zum Zugriff auf andere Objekte.

Vorgehensweise

1. Erstellen Sie die Benutzer-ID *mqttUser* als Identität aller MQTT-Clients.
2. Berechtigen Sie *mqttUser* zum Veröffentlichen und Subskribieren aller Themen und zum Senden von Veröffentlichungen an MQTT-Clients.

```
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
```

Mittlere Zugriffssteuerung
MQTT-Clients sind in verschiedene Gruppen unterteilt, um verschiedene Themengruppen zu veröffentlichen und zu abonnieren und Nachrichten an MQTT-Clients zu senden.

Vorgehensweise

1. Erstellen Sie mehrere Benutzer-IDs, *mqttUsers* und mehrere Verwaltungsthemen in der Publish/Subscribe-Themenstruktur.
2. Autorisieren Sie verschiedene *mqttUsers* zu verschiedenen Themen.

```
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. Erstellen Sie eine Gruppe *mqtt* und fügen Sie alle *mqttUsers* zur Gruppe hinzu.
4. Berechtigen Sie *mqtt* zum Senden von Topics an MQTT-Clients.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Feinkörnige Zugriffssteuerung
MQTT-Clients werden in ein vorhandenes System der Zugriffssteuerung integriert, das Gruppen berechtigt, Aktionen für Objekte auszuführen.

Informationen zu diesem Vorgang

Eine Benutzer-ID wird abhängig von den Berechtigungen, die sie benötigt, einer oder mehreren Betriebssystemgruppen zugeordnet. Verwenden Sie dieses Modell, wenn WebSphere MQ -Anwendungen denselben Topicbereich wie MQTT-Clients veröffentlichen und subskribieren. Die Gruppen werden als Publish*X*, Subscribe*Y*und `mqtt` bezeichnet

Publish*X*
Mitglieder von Publish*X* -Gruppen können in *topicX*veröffentlichen.

Subscribe*Y*
Mitglieder von Subscribe*Y* -Gruppen können *topicY*abonnieren.

`mqtt`
Mitglieder der Gruppe *`mqtt`* können Veröffentlichungen an MQTT-Clients senden.

Vorgehensweise

1. Erstellen Sie mehrere Gruppen Publish*X* und Subscribe*Y* , die mehreren Verwaltungsthemen in der Publish/Subscribe-Themenstruktur zugeordnet sind.
2. Erstellen Sie eine Gruppe `mqtt`.
3. Erstellen Sie mehrere Benutzer-IDs, *`mqttUsers`*, und fügen Sie die Benutzer zu einer der Gruppen hinzu, je nachdem, welche Berechtigung sie haben, um sie zu tun.
4. Berechtigen Sie verschiedene Gruppen Publish*X* und Subscribe*X* für unterschiedliche Topics und berechtigen Sie die Gruppe *`mqtt`* für das Senden von Nachrichten an MQTT-Clients.

```
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
```

MQTT-Clientauthentifizierung mithilfe eines Kennworts

Authentifizieren Sie den `Benutzernamen` mithilfe des Clientkennworts. Sie können den Client mit einer anderen Identität gegenüber der Identität authentifizieren, die verwendet wird, um den Client zu autorisieren, Themen zu veröffentlichen und zu subskribieren.

Der Telemetrieservice (MQXR) verwendet JAAS, um den Client `benutzername` zu authentifizieren. JAAS verwendet das vom MQTT-Client übergebene `Kennwort`.

Der WebSphere MQ-Administrator entscheidet, ob der `Benutzername` authentifiziert wird oder ob überhaupt keine Authentifizierung stattfindet. Hierfür konfiguriert er den MQTT-Kanal, mit dem sich ein Client verbindet. Clients können verschiedenen Kanälen zugeordnet werden, und jeder Kanal kann so konfiguriert werden, dass er seine Clients auf unterschiedliche Weise authentifiziert. Mit JAAS können Sie konfigurieren, welche Methoden den Client authentifizieren müssen und die optional den Client authentifizieren können.

Die Auswahl der Identität für die Authentifizierung hat keinen Einfluss auf die Auswahl der Identität für die Berechtigung. Es kann sein, dass Sie eine gemeinsame Identität für die Berechtigung für den Verwaltungskomfort einrichten möchten, aber jeder Benutzer authentifiziert wird, um diese Identität zu verwenden. In der folgenden Prozedur werden die Schritte zur Authentifizierung einzelner Benutzer für die Verwendung einer gemeinsamen Identität beschrieben:

1. Der WebSphere MQ-Administrator legt mithilfe von WebSphere MQ Explorer für die MQTT-Kanalidentität einen beliebigen Namen fest (z. B. `MQTTClientUser`).
2. Der WebSphere MQ-Administrator autorisiert `MQTTClient` für das Veröffentlichen und Subskribieren von allen Themen:

```
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. Der MQTT-Clientanwendungsentwickler erstellt ein `MqttConnectOptions` -Objekt und legt `Benutzername` und `Kennwort` fest, bevor er eine Verbindung zum Server herstellt.

4. Der Sicherheitsentwickler erstellt ein JAAS `LoginModule` zur Authentifizierung des `Benutzernamens` mit dem `Kennwort` und schließt es in die JAAS -Konfigurationsdatei ein.
5. Der WebSphere MQ-Administrator konfiguriert unter Verwendung von JAAS den MQTT-Kanal für die Authentifizierung des Benutzernamens (`UserName`) des Clients.

MQTT-Clientauthentifizierung mithilfe von SSL

Verbindungen zwischen dem MQTT-Client und dem Warteschlangenmanager werden immer vom MQTT-Client eingeleitet. Der MQTT-Client ist immer der SSL-Client. Die Clientauthentifizierung für den Server und die Serverauthentifizierung des MQTT-Clients sind beide optional.

Wenn Sie dem Client ein privates signiertes digitales Zertifikat bereitstellen, können Sie den MQTT-Client bei IBM WebSphere MQauthentifizieren. Der IBM WebSphere MQ-Administrator kann MQTT-Clients auch zwingen, sich selbst über SSL beim Warteschlangenmanager zu authentifizieren. Eine Clientauthentifizierung kann nur im Rahmen einer gegenseitigen Authentifizierung angefordert werden.

Als Alternative zu SSL können bestimmte virtuelle private Netze (Virtual Private Network, VPN), z. B. IPsec, die Endpunkte einer TCP/IP-Verbindung authentifizieren. In einem VPN wird jedes IP-Paket, das im Netz übertragen wird, verschlüsselt. Sobald eine solche VPN-Verbindung hergestellt ist, haben Sie ein vertrauenswürdiges Netz aufgebaut. MQTT-Clients können auch mit TCP/IP über das VPN mit Telemetriekanälen verbunden werden.

Die Clientauthentifizierung über SSL setzt voraus, dass der Client über einen geheimen Schlüssel verfügt. Der geheime Schlüssel ist im Falle eines selbst signierten Zertifikats der private Schlüssel des Clients bzw. andernfalls ein Schlüssel, der von einer Zertifizierungsstelle ausgestellt wird. Mit dem Schlüssel wird das digitale Zertifikat des Clients signiert. Jede Person im Besitz des entsprechenden öffentlichen Schlüssels kann das digitale Zertifikat verifizieren. Zertifikate können vertrauenswürdig sein oder - falls sie verkettet sind - durch eine Zertifikatskette bis zu einem Trusted-Root-Zertifikat zurückverfolgt werden. Bei der Clientverifizierung werden alle Zertifikate in der vom Client bereitgestellten Zertifikatskette an den Server gesendet. Der Server überprüft die Zertifikatskette, bis er ein Zertifikat findet, dem er vertraut. Das vertrauenswürdige Zertifikat ist entweder das öffentliche Zertifikat, das auf Basis eines selbst signierten Zertifikats generiert wird, oder ein Stammzertifikat, das für gewöhnlich von einer Zertifizierungsstelle ausgestellt wird. In einem letzten (optionalen) Schritt kann das vertrauenswürdige Zertifikat mit einer aktuellen Zertifikatswiderrufsliste abgeglichen werden.

Das vertrauenswürdige Zertifikat kann von einer Zertifizierungsstelle ausgegeben und bereits im JRE-Zertifikatsspeicher enthalten sein. Es kann sich aber auch um ein selbst signiertes Zertifikat oder um jedes Zertifikat handeln, das dem Keystore des Telemetriekanals als vertrauenswürdiges Zertifikat hinzugefügt wurde.

Anmerkung: Der Telemetriekanal verfügt über eine Kombination aus Keystore/Truststore, die sowohl private Schlüssel für einen oder mehrere Telemetriekanäle enthält, als auch öffentliche Zertifikate, die zur Authentifizierung von Clients erforderlich sind. Da ein SSL-Kanal einen Keystore haben muss und dies dieselbe Datei wie der Truststore des Kanals ist, wird der JRE-Zertifikatsspeicher niemals referenziert. Dies führt dazu, dass Sie das Stammzertifikat in den Keystore für den Kanal stellen müssen, wenn für eine Authentifizierung eines Clients ein Stammzertifikat einer Zertifizierungsstelle erforderlich ist. Dies gilt selbst dann, wenn sich das Stammzertifikat der Zertifizierungsstelle bereits im JRE-Zertifikatsspeicher befindet. Der JRE-Zertifikatsspeicher wird niemals referenziert.

Überlegen Sie sich, welchen Sicherheitsbedrohungen die Clientauthentifizierung entgegenwirken soll, und welche Rollen der Client und Server bei der Abwendung der Sicherheitsbedrohungen spielen. Die Authentifizierung des Clientzertifikats reicht allein nicht aus, um unbefugten Zugriff auf ein System zu verhindern. Wenn eine dritte Person über das Clientgerät verfügt, handelt das Clientgerät nicht unbedingt mit der Berechtigung des Zertifikatseigners. Verlassen Sie sich bei der Abwehr von unerwünschten Attacken niemals auf eine einzige Maßnahme. Verwenden Sie mindestens eine aus zwei Faktoren bestehende Authentifizierung und sichern Sie dies durch den Besitz eines Zertifikats ab, für das nicht öffentliche Informationen bekannt sein müssen. Verwenden Sie beispielsweise JAAS und authentifizieren Sie den Client mittels eines vom Server ausgestellten Kennworts.

Die größte Sicherheitsbedrohung für das Clientzertifikat ist, dass es in falsche Hände gelangt. Das Zertifikat befindet sich in einem kennwortgeschützten Keystore beim Client. Wie wird es in den Keystore

gestellt? Wie erhält der MQTT-Client das Kennwort für den Keystore? Wie sicher ist der Kennwortschutz? Telemetriegeräte sind häufig einfach zu entfernen und können dann in Ruhe manipuliert werden. Muss das Gerät manipulationssicher sein? Die Verteilung und der Schutz clientseitiger Zertifikate gilt als besonders schwierig; diese Aufgabe wird als das Schlüsselverwaltungsproblem bezeichnet.

Eine weitere große Bedrohung besteht darin, dass das Gerät unbeabsichtigt für den Zugriff auf Server missbraucht wird. Wird die MQTT-Anwendung beispielsweise manipuliert, kann unter Verwendung der authentifizierten Clientidentität eine Schwachstelle in der Serverkonfiguration ausgenutzt werden.

Zur Authentifizierung eines MQTT-Clients mit SSL müssen Sie den Telemetriekanal und den Client konfigurieren.

•
•

Konfiguration des Telemetriekanals für MQTT-Clientauthentifizierung mit SSL

Der IBM WebSphere MQ -Administrator konfiguriert Telemetriekanäle auf dem Server. Jeder Kanal wird so konfiguriert, dass er eine TCP/IP-Verbindung an einer anderen Portnummer akzeptiert. SSL-Kanäle werden mit kennphrasengeschütztem Zugriff auf Schlüsseldateien konfiguriert. Wird ein SSL-Kanal ohne Kennphrase oder Schlüsseldatei definiert, akzeptiert der Kanal keine SSL-Verbindungen.

Setzen Sie die Eigenschaft `com.ibm.mq.MQTT.ClientAuth` eines SSL-Telemetriekanals auf `REQUIRED`, um alle Clients, die eine Verbindung mit dem Kanal herstellen, zu zwingen, den Nachweis zu erbringen, dass sie verifizierte digitale Zertifikate besitzen. Die Clientzertifikate werden mit Hilfe von Zertifikaten von Zertifizierungsstellen authentifiziert, was zu einem Trusted-Root-Zertifikat führt. Wenn das Clientzertifikat selbst signiert ist oder von einem Zertifikat signiert wird, das von einer Zertifizierungsstelle ausgestellt wurde, müssen die öffentlich signierten Zertifikate der Client-oder Zertifizierungsstelle sicher auf dem Server gespeichert werden.

Stellen Sie das öffentlich signierte Clientzertifikat oder das Zertifikat aus der Zertifizierungsstelle in den Telemetriekanal-Keystore. Auf dem Server werden öffentlich signierte Zertifikate in derselben Schlüsseldatei wie privat signierte Zertifikate gespeichert, und nicht in einem separaten Truststore.

Der Server überprüft die Signatur aller Clientzertifikate, die er mit allen öffentlichen Zertifikaten und Cipher Suites, die er hat, gesendet wird. Der Server überprüft die Schlüsselkette. Der WS-Manager kann so konfiguriert werden, dass er das Zertifikat gegen die Zertifikatswiderrufsliste testen kann. Die Eigenschaft für die Widerrufsnamensliste des Warteschlangenmanagers heißt `SSLCRLNL`.

Wenn eines der Zertifikate, die ein Client sendet, durch ein Zertifikat im Serverschlüsselspeicher verifiziert wird, wird der Client authentifiziert.

Der WebSphere MQ-Administrator kann den gleichen Telemetriekanal auch so konfigurieren, dass er JAAS zur Überprüfung von `UserName` bzw. `ClientIdentifier` des Clients mit dem Client-`Password` verwendet.

Sie können denselben Schlüsselspeicher für mehrere Telemetriekanäle verwenden.

Die Prüfung von mindestens einem digitalen Zertifikat im kennwortgeschützten Client-Keystore auf dem Gerät authentifiziert den Client gegenüber dem Server. Das digitale Zertifikat wird nur zur Authentifizierung durch WebSphere MQ verwendet, Es wird nicht verwendet, um die TCP/IP-Adresse des Clients zu überprüfen, oder die Identität des Clients für die Autorisierung oder Abrechnung festzulegen. Die auf dem Server wahrgenommene Identität des Clients ist entweder sein `UserName` oder sein `ClientIdentifier` bzw. eine entsprechende vom WebSphere MQ-Administrator eingerichtete Identität.

Sie können für die Clientauthentifizierung auch SSL-Cipher-Suites verwenden.Es folgt eine alphabetische Liste der SSL Cipher-Suites, die momentan unterstützt werden:

- `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

V7.5.0.2 Wenn SHA-2-Cipher-Suites verwendet werden sollen, lesen Sie den Abschnitt Systemvoraussetzungen zur Verwendung von SHA-2-Cipher-Suites mit MQTT-Kanälen.

Zugehörige Konzepte

„Konfiguration des Telemetriekanals für Kanalauthentifizierung mit SSL” auf Seite 136

Der IBM WebSphere MQ -Administrator konfiguriert Telemetriekanäle auf dem Server. Jeder Kanal wird so konfiguriert, dass er eine TCP/IP-Verbindung an einer anderen Portnummer akzeptiert. SSL-Kanäle

werden mit kennphrasengeschütztem Zugriff auf Schlüsseldateien konfiguriert. Wird ein SSL-Kanal ohne Kennphrase oder Schlüsseldatei definiert, akzeptiert der Kanal keine SSL-Verbindungen.

CipherSpecs und CipherSuites

Zugehörige Verweise

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

Authentifizierung des Telemetriekanals über SSL

Verbindungen zwischen dem MQTT-Client und dem Warteschlangenmanager werden immer vom MQTT-Client eingeleitet. Der MQTT-Client ist immer der SSL-Client. Die Clientauthentifizierung für den Server und die Serverauthentifizierung des MQTT-Clients sind beide optional.

Der Client versucht immer, den Server zu authentifizieren, es sei denn, der Client wurde für die Verwendung einer CipherSpec konfiguriert, die anonyme Verbindungen unterstützt. Schlägt die Authentifizierung fehl, wird keine Verbindung hergestellt.

Als Alternative zu SSL können bestimmte virtuelle private Netze (Virtual Private Network, VPN), z. B. IPsec, die Endpunkte einer TCP/IP-Verbindung authentifizieren. In einem VPN wird jedes IP-Paket, das im Netz übertragen wird, verschlüsselt. Sobald eine solche VPN-Verbindung hergestellt ist, haben Sie ein vertrauenswürdiges Netz aufgebaut. MQTT-Clients können auch mit TCP/IP über das VPN mit Telemetriekanälen verbunden werden.

Bei der Serverauthentifizierung mit SSL wird der Server, an den Sie vertrauliche Informationen senden möchten, mithilfe von Secure Sockets Layer (SSL) authentifiziert. Der Client führt die Prüfungen durch, die mit den vom Server gesendeten Zertifikaten übereinstimmen, mit Zertifikaten, die in seinem Truststore oder in seinem JRE- `cacerts` Speicher gespeichert sind.

Der JRE-Zertifikatsspeicher ist eine JKS-Datei, `cacerts`. Sie befindet sich im Pfad `JRE Install-Path\lib\security\`. Sie wird mit dem Standardkennwort `changeit` installiert. Sie können Zertifikate, denen Sie vertrauen, entweder im JRE-Zertifikatsspeicher oder im Truststore des Clients speichern. Sie können jedoch nicht beide Speicher verwenden. Verwenden Sie den Truststore des Clients, wenn Sie die öffentlichen Zertifikate, denen der Client vertraut, getrennt von den Zertifikaten aufbewahren möchten, die von anderen Java-Anwendungen verwendet werden. Verwenden Sie den JRE-Zertifikatsspeicher, wenn Sie einen allgemeinen Zertifikatsspeicher für alle Java-Anwendungen nutzen möchten, die auf dem Client ausgeführt werden. Wenn Sie sich für die Verwendung des JRE-Zertifikatsspeichers entscheiden, überprüfen Sie die darin enthaltenen Zertifikate, um sicherzustellen, dass diese wirklich vertrauenswürdig sind.

Sie können die JSSE-Konfiguration ändern, indem Sie einen anderen Trust-Provider bereitstellen. Ein Trust-Provider kann so angepasst werden, dass er verschiedene Zertifikatsprüfungen durchführt. In einigen OGSi-Umgebungen, die den MQTT-Client verwendet haben, stellt die Umgebung einen unterschiedlichen Trust-Provider bereit.

Zur Authentifizierung eines Telemetriekanals mit SSL müssen Sie den Server und den Client konfigurieren.

-
-

Konfiguration des Telemetriekanals für Kanalauthentifizierung mit SSL

Der IBM WebSphere MQ -Administrator konfiguriert Telemetriekanäle auf dem Server. Jeder Kanal wird so konfiguriert, dass er eine TCP/IP-Verbindung an einer anderen Portnummer akzeptiert. SSL-Kanäle werden mit kennphrasengeschütztem Zugriff auf Schlüsseldateien konfiguriert. Wird ein SSL-Kanal ohne Kennphrase oder Schlüsseldatei definiert, akzeptiert der Kanal keine SSL-Verbindungen.

Speichern Sie das digitale Zertifikat des Servers, signiert mit seinem privaten Schlüssel, im Keystore, den der Telemetriekanal auf dem Server verwenden wird. Speichern Sie alle Zertifikate in seiner Schlüsselkette im Keystore, wenn Sie die Schlüsselkette an den Client übertragen möchten. Konfigurieren Sie den Telemetriekanal mit WebSphere MQ Explorer für die Verwendung von SSL. Geben Sie den Pfad zum Keystore und den Verschlüsselungstext für den Zugriff auf den Schlüsselspeicher an. Wenn Sie die

TCP/IP-Portnummer des Kanals nicht festlegen, wird standardmäßig 8883 als Portnummer des SSL-Telemetriekanals verwendet.

Alternativ können für die Kanalauthentifizierung auch SSL Cipher-Suites verwendet werden.Es folgt eine alphabetische Liste der SSL Cipher-Suites, die momentan unterstützt werden:

- 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

- V 7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V 7.5.0.2 Wenn SHA-2-Cipher-Suites verwendet werden sollen, lesen Sie den Abschnitt Systemvoraussetzungen zur Verwendung von SHA-2-Cipher-Suites mit MQTT-Kanälen.

Zugehörige Konzepte

„Konfiguration des Telemetriekanals für MQTT-Clientauthentifizierung mit SSL" auf Seite 134
Der IBM WebSphere MQ -Administrator konfiguriert Telemetriekanäle auf dem Server. Jeder Kanal wird so konfiguriert, dass er eine TCP/IP-Verbindung an einer anderen Portnummer akzeptiert. SSL-Kanäle werden mit kennphrasengeschütztem Zugriff auf Schlüsseldateien konfiguriert. Wird ein SSL-Kanal ohne Kennphrase oder Schlüsseldatei definiert, akzeptiert der Kanal keine SSL-Verbindungen.

CipherSpecs und CipherSuites

Zugehörige Verweise

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)

Datenschutz auf Telemetriekanälen

Die Vertraulichkeit von MQTT-Veröffentlichungen, die in beiden Richtungen über Telemetriekanäle gesendet werden, wird dadurch sichergestellt, dass Übertragungen über die Verbindung mit SSL verschlüsselt werden.

MQTT-Clients, die Verbindungen mit Telemetriekanälen herstellen, verwenden SSL, um die Vertraulichkeit von Veröffentlichungen, die über die Kanäle übertragen werden, durch eine symmetrische Verschlüsselung sicherzustellen. Da die Endpunkte nicht authentifiziert sind, können Sie die Kanalverschlüsselung nicht allein vertrauen. Kombinieren Sie den Schutz der Privatsphäre mit dem Server oder der gegenseitigen Authentifizierung.

Als Alternative zu SSL können bestimmte virtuelle private Netze (Virtual Private Network, VPN), z. B. IPsec, die Endpunkte einer TCP/IP-Verbindung authentifizieren. In einem VPN wird jedes IP-Paket, das im Netz übertragen wird, verschlüsselt. Sobald eine solche VPN-Verbindung hergestellt ist, haben Sie ein vertrauenswürdiges Netz aufgebaut. MQTT-Clients können auch mit TCP/IP über das VPN mit Telemetriekanälen verbunden werden.

Informationen zu einer typischen Konfiguration mit Kanalverschlüsselung und Serverauthentifizierung finden Sie im Abschnitt „Authentifizierung des Telemetriekanals über SSL" auf Seite 136.

Wird eine SSL-Verbindung ohne Authentifizierung des Servers verschlüsselt, ist die Verbindung für Man-in-the-Middle-Attacken anfällig. Obwohl die Informationen, die Sie austauschen, vor Lauschangriffen geschützt sind, wissen Sie nicht, mit wem Sie sie austauschen. Wenn Sie das Netz nicht steuern, sind Sie einem Benutzer ausgesetzt, der Ihre IP-Übertragungen abfängt und als Endpunkt maskiert wird.

Sie können eine verschlüsselte SSL-Verbindung aufbauen (ohne Authentifizierung des Servers), indem Sie eine Diffie-Hellman-Schlüsselaustausch-CipherSpec, die anonymes SSL unterstützt, einsetzen. Der geheime Master-Schlüssel, den Client und Server gemeinsam nutzen und mit dem SSL-Übertragungen verschlüsselt werden, wird erstellt, ohne dass ein privat signiertes Serverzertifikat ausgetauscht wird.

Da anonyme Verbindungen unsicher sind, verwenden SSL-Implementierungen standardmäßig keine anonymen CipherSpecs. Wenn eine Clientanforderung für eine SSL-Verbindung von einem Telemetriekanal akzeptiert wird, muss der Kanal über einen durch eine Kennphrase geschützten Schlüsselspeicher verfügen. Da SSL-Implementierungen keine anonymen CipherSpecs verwenden, muss der Schlüsselspeicher standardmäßig ein privat signiertes Zertifikat enthalten, das der Client authentifizieren kann.

Wenn Sie anonyme CipherSpecs verwenden, muss der Server-Keystore vorhanden sein, aber er muss keine privat signierten Zertifikate enthalten.

Eine weitere Möglichkeit, eine verschlüsselte Verbindung herzustellen, besteht darin, den Trust-Provider auf dem Client durch Ihre eigene Implementierung zu ersetzen. Ihr Trust-Provider würde das Serverzertifikat nicht authentifizieren, aber die Verbindung würde verschlüsselt sein.

SSL-Konfiguration der MQTT-Clients und Telemetriekanäle

MQT-Clients und der WebSphere MQ Telemetry -Service (MQXR) verwenden Java Secure Socket Extension (JSSE), um Telemetriekanäle über SSL zu verbinden. Der IBM WebSphere MQ Telemetry-Dämon für Geräte unterstützt SSL nicht.

Konfigurieren Sie SSL, um den Telemetriekanal und den MQTT-Client zu authentifizieren und die Übertragung von Nachrichten zwischen Clients und dem Telemetriekanal zu verschlüsseln.

Als Alternative zu SSL können bestimmte virtuelle private Netze (Virtual Private Network, VPN), z. B. IPsec, die Endpunkte einer TCP/IP-Verbindung authentifizieren. In einem VPN wird jedes IP-Paket, das im Netz übertragen wird, verschlüsselt. Sobald eine solche VPN-Verbindung hergestellt ist, haben Sie ein vertrauenswürdiges Netz aufgebaut. MQTT-Clients können auch mit TCP/IP über das VPN mit Telemetriekanälen verbunden werden.

Sie können die Verbindung zwischen einem Java-MQTT-Client und einem Telemetriekanal für die Verwendung des SSL-Protokolls über TCP/IP konfigurieren. Was gesichert wird, hängt davon ab, wie Sie SSL für die Verwendung von JSSE konfigurieren. Beginnend mit der sichersten Konfiguration können Sie drei verschiedene Sicherheitsstufen konfigurieren:

1. Erlauben Sie nur vertrauenswürdigen MQTT-Clients die Herstellung einer Verbindung. Verbinden Sie einen MQTT-Client nur mit einem vertrauenswürdigen Telemetriekanal. Verschlüsseln Sie Nachrichten zwischen dem Client und dem Warteschlangenmanager. Weitere Informationen finden Sie im Abschnitt „MQTT-Clientauthentifizierung mithilfe von SSL” auf Seite 133.
2. Verbinden Sie einen MQTT-Client nur mit einem vertrauenswürdigen Telemetriekanal. Verschlüsseln Sie Nachrichten zwischen dem Client und dem Warteschlangenmanager; siehe „Authentifizierung des Telemetriekanals über SSL” auf Seite 136.
3. Verschlüsseln Sie Nachrichten zwischen dem Client und dem Warteschlangenmanager; siehe „Datenschutz auf Telemetriekanälen” auf Seite 138.

JSSE-Konfigurationsparameter

Durch die Modifizierung der JSSE-Parameter können Sie die Art ändern, wie eine SSL-Verbindung konfiguriert ist. Die JSSE-Konfigurationsparameter sind in drei Gruppen angeordnet:

1. IBM WebSphere MQ Telemetriekanal
2. MQTT-Java-Client
3. JRE

Konfigurieren Sie die Telemetriekanalparameter mithilfe von IBM WebSphere MQ Explorer. Legen Sie die MQTT-Java-Clientparameter im Attribut `MqttConnectionOptions.SSLProperties` fest. Ändern Sie die JRE-Sicherheitsparameter, indem Sie Dateien im JRE-Sicherheitsverzeichnis sowohl auf dem Client als auch auf dem Server bearbeiten.

IBM WebSphere MQ Telemetry-Kanal

Legen Sie alle SSL-Parameter des Telemetriekanals mithilfe von WebSphere MQ Explorer fest.

`ChannelName`

`ChannelName` ist ein erforderlicher Parameter auf allen Kanälen.

Der Kanalname identifiziert den Kanal, der einer bestimmten Port-Nummer zugeordnet ist. Geben Sie den Kanälen einen Namen, um die Verwaltung von MQTT-Clientgruppen zu vereinfachen.

`PortNumber`

`PortNumber` ist ein optionaler Parameter auf allen Kanälen. Bei TCP-Kanälen hat er standardmäßig den Wert 1883, bei SSL-Kanälen den Wert 8883.

Die TCP/IP-Anschlussnummer, die diesem Kanal zugeordnet ist. MQTT-Clients werden durch die Angabe des Ports verbunden, der für den Kanal definiert ist. Wenn der Kanal über SSL-Eigenschaften verfügt, muss sich der Client unter Verwendung des SSL-Protokolls verbinden. Beispiel:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

KeyFileName ist ein erforderlicher Parameter für SSL-Kanäle. Er muss für TCP-Kanäle weggelassen werden.

KeyFileName ist der Pfad zum Java-Keystore mit den digitalen Zertifikaten, die Sie angeben. Verwenden Sie JKS, JCEKS oder PKCS12 als den Typ des Schlüsselspeichers auf dem Server.

Geben Sie den Keystore-Typ mithilfe einer der folgenden Dateierweiterungen an:

```
.jks
.jceks
.p12
.pkcs12
```

Es wird angenommen, dass ein Schlüsselspeicher mit einer anderen Dateierweiterung ein JKS-Keystore ist.

Sie können einen Typ von Schlüsselspeicher auf dem Server mit anderen Keystoretypen auf dem Client kombinieren.

Speichern Sie das private Zertifikat des Servers in den Schlüsselspeicher. Das Zertifikat wird als Serverzertifikat bezeichnet. Das Zertifikat kann selbst signiert sein oder Teil einer Zertifikatskette sein, die von einer Signaturberechtigung signiert ist.

Wenn Sie eine Zertifikatskette verwenden, legen Sie die zugeordneten Zertifikate in den Serverschlüsselspeicher.

Das Serverzertifikat und alle Zertifikate in seiner Zertifikatskette werden an Clients gesendet, um die Identität des Servers zu authentifizieren.

Wenn Sie ClientAuth auf Required gesetzt haben, muss der Keystore alle Zertifikate enthalten, die für die Authentifizierung des Clients erforderlich sind. Der Client sendet ein selbst signiertes Zertifikat oder eine Zertifikatskette, und der Client wird durch die erste Überprüfung dieses Materials anhand eines Zertifikats im Keystore authentifiziert. Mit Hilfe einer Zertifikatskette kann ein Zertifikat viele Clients prüfen, auch wenn sie mit unterschiedlichen Clientzertifikaten ausgegeben werden.

PassPhrase

PassPhrase ist ein erforderlicher Parameter für SSL-Kanäle. Er muss für TCP-Kanäle weggelassen werden.

Der Verschlüsselungstext wird zum Schutz des Keystores verwendet.

ClientAuth

ClientAuth ist ein optionaler SSL-Parameter. Der Standardwert ist keine Clientauthentifizierung. Er muss für TCP-Kanäle weggelassen werden.

Legen Sie ClientAuth fest, wenn der Telemetrieservice (MQXR) den Client authentifizieren soll, bevor er dem Client die Verbindung zum Telemetriekanal ermöglicht.

Wenn Sie ClientAuth festlegen, muss sich der Client unter Verwendung von SSL mit dem Server verbinden und den Server authentifizieren. Als Antwort auf die Einstellung von ClientAuth sendet der Client sein digitales Zertifikat an den Server und alle anderen Zertifikate in seinem Keystore. Das digitale Zertifikat wird als Clientzertifikat bezeichnet. Diese Zertifikate werden für Zertifikate authentifiziert, die im Channel-Keystore und im JRE- cacerts speicher gespeichert sind.

CipherSuite

`CipherSuite` ist ein optionaler SSL-Parameter. Standardmäßig werden alle aktivierten CipherSpecs probiert. Er muss für TCP-Kanäle weggelassen werden.

Wenn Sie eine bestimmte CipherSpec verwenden möchten, setzen Sie `CipherSuite` auf den Namen der CipherSpec, die für die Herstellung der SSL-Verbindung verwendet werden muss.

Der Telemetrieservice und der MQTT-Client handeln eine gemeinsame CipherSpec aus, die aus allen CipherSpecs gewählt wird, die auf jeder Seite aktiviert sind. Wenn eine bestimmte CipherSpec an einem der beiden oder beiden Enden der Verbindung angegeben ist, muss sie mit der CipherSpec am anderen Ende übereinstimmen.

Installieren Sie zusätzliche Chiffriergeräte, indem Sie JSSE zusätzliche Provider hinzufügen.

Federal Information Processing Standards (FIPS)

FIPS ist eine optionale Einstellung. Standardmäßig ist sie nicht festgelegt.

Legen Sie entweder in der Eigenschaftenanzeige des Warteschlangenmanagers oder mithilfe von **runmqsc** SSLFIPSfest. `SSLFIPS` gibt an, ob nur FIPS-zertifizierte Algorithmen verwendet werden sollen.

Revocation namelist

Die Widerrufsnamensliste ist eine optionale Einstellung. Standardmäßig ist sie nicht festgelegt.

Legen Sie SSLCRLNLin der Eigenschaftenanzeige des Warteschlangenmanagers oder mithilfe von **runmqsc**fest. `SSLCRLNL` gibt eine Namensliste mit Authentifizierungsinformationsobjekten an, die zum Angeben von Zertifikatswiderrufspositionen verwendet werden.

Es werden keine anderen Warteschlangenmanager-Parameter verwendet, die SSL-Eigenschaften festlegen.

MQTT-Java-Client

Legen Sie SSL-Eigenschaften für den Java-Client in `MqttConnectionOptions.SSLProperties`fest. Beispiel:

```
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);
```

Die Namen und Werte bestimmter Eigenschaften werden in der Klasse `MqttConnectOptions` beschrieben. Links zur Client-API-Dokumentation für die MQTT-Clientbibliotheken finden Sie unter MQTT client programming reference.

Protocol

Das `Protokoll` ist optional.

Das Protokoll wird in den Verhandlungen mit dem Telemetrieserver ausgewählt. Wenn Sie ein bestimmtes Protokoll benötigen, können Sie eine auswählen. Wenn der Telemetrieserver das Protokoll nicht unterstützt, schlägt die Verbindung fehl.

ContextProvider

`ContextProvider` ist optional.

KeyStore

`KeyStore` ist optional. Konfigurieren Sie sie, wenn `ClientAuth` auf dem Server festgelegt ist, um die Authentifizierung des Clients zu erzwingen.

Speichern Sie das digitale Zertifikat des Clients, signiert mit seinem privaten Schlüssel, in den Schlüsselspeicher. Geben Sie den Schlüsselspeicherpfad und das Kennwort an. Der Typ und der Provider sind optional. JKS ist der Standardtyp, und IBMJCE ist der Standardprovider.

Geben Sie einen anderen Schlüsselspeicherprovider an, um auf eine Klasse zu verweisen, die einen neuen Keystore-Provider hinzufügt Übergeben Sie den Namen des Algorithmus, der vom Schlüsselspeicherprovider für die Instanziierung von KeyManagerFactory verwendet wird, indem Sie den Namen des Schlüsselmanagers festlegen.

TrustStore

TrustStore ist optional. Sie können alle Zertifikate, auf die Sie vertrauen, in den JRE- cacerts speicher stellen.

Konfigurieren Sie den Truststore, wenn Sie einen anderen Truststore für den Client verwenden möchten. Sie können den Truststore möglicherweise nicht konfigurieren, wenn der Server ein Zertifikat verwendet, das von einer bekannten Zertifizierungsinstanz ausgestellt wurde, die bereits sein Stammzertifikat in cacerts gespeichert hat.

Fügen Sie das öffentlich signierte Zertifikat des Servers oder des Stammzertifikats zum Truststore hinzu, und geben Sie den Truststore-Pfad und das Kennwort an. JKS ist der Standardtyp, und IBMJCE ist der Standardprovider.

Geben Sie einen anderen Truststore-Provider an, um auf eine Klasse zu verweisen, die einen neuen Truststore-Provider hinzufügt. Übergeben Sie den Namen des Algorithmus, der vom Truststore-Provider für die Instanziierung der TrustManagerFactory verwendet wird, indem Sie den Namen des Trust-Managers festlegen.

JRE

Sonstige Aspekte der Java-Sicherheit, die sich auf das SSL-Verhalten auf dem Client und dem Server auswirken, werden in der Java Runtime Environment (JRE) konfiguriert. Die Konfigurationsdateien unter Windows befinden sich im Verzeichnis *Java Installation Directory*\jre\lib\security. Wenn Sie die JRE verwenden, die mit IBM WebSphere MQ ausgeliefert wird, lautet der Pfad wie in der folgenden Tabelle angegeben:

Tabelle 8. Dateipfade nach Plattform für JRE-SSL-Konfigurationsdateien

Plattform	**Dateipfad**
Windows	*WMQ Installation Directory*\java\jre\lib\security
Sonstige UNIX and Linux-Plattformen	*WMQ Installation Directory*/java/jre64/jre/lib/security

Well-bekannte Zertifizierungsstellen

Die Datei cacerts enthält die Stammzertifikate anerkanster Zertifizierungsstellen. Der cacerts wird standardmäßig verwendet, es sei denn, Sie geben einen Truststore an. Wenn Sie den cacerts -Store verwenden oder keinen Truststore bereitstellen, müssen Sie die Liste der Unterzeichner in cacerts überprüfen und bearbeiten, um Ihre Sicherheitsanforderungen zu erfüllen.

Sie können cacerts mit dem WebSphere MQ-Befehl strmqikm öffnen. Dieser Befehl führt das Dienstprogramm IBM Key Management aus. Öffnen Sie cacerts als JKS-Datei, indem Sie das Kennwort changeit verwenden. Ändern Sie das Kennwort, um die Datei zu sichern.

Sicherheitsklassen konfigurieren

Verwenden Sie die Datei java.security , um zusätzliche Sicherheitsprovider und andere Standardsicherheitseigenschaften zu registrieren.

Zulassungs-
Verwenden Sie die Datei java.policy , um die Berechtigungen zu ändern, die für Ressourcen erteilt wurden. javaws.policy erteilt Berechtigungen für javaws.jar.

Verschlüsselungsstärke
Einige JREs werden mit reduzierter Verschlüsselungsstärke ausgeliefert. Wenn Sie keine Schlüssel in Keystores importieren können, kann die Verschlüsselung mit verminderter Stärke die Ursache sein. Versuchen Sie entweder, **ikeyman** mit dem Befehl **strmqikm** zu starten, oder laden Sie

starke, aber eingeschränkte Jurisdiction-Dateien von der Website IBM developer kits, Security information herunter.

Wichtig: Ihr Herkunftsland hat möglicherweise Einschränkungen für den Import, den Besitz, die Verwendung oder den erneuten Export in ein anderes Land der Verschlüsselungssoftware zur Verfügung. Bevor Sie die unbeschränkten Richtliniendateien herunterladen oder verwenden, müssen Sie die Gesetze Ihres Landes überprüfen. Überprüfen Sie die Bestimmungen und die Richtlinien für den Import, den Besitz, die Verwendung und den Wiederexport von Verschlüsselungssoftware, um festzustellen, ob die Software zugelassen ist.

Ändern Sie den Trust-Provider, damit der Client eine Verbindung zu einem beliebigen Server herstellen kann.

Das Beispiel veranschaulicht, wie Sie einen Trust-Provider hinzufügen und vom MQTT-Client-Code aus referenzieren können. Das Beispiel führt keine Authentifizierung für den Client oder den Server durch. Die hergestellte SSL-Verbindung ist verschlüsselt, aber nicht authentifiziert.

In dem Codeausschnitt in Abbildung 25 auf Seite 143 werden der Trust-Provider `AcceptAllProviders` und der Trust-Manager für den MQTT-Client festgelegt.

```
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);
```

Abbildung 25. Codeausschnitt des MQTT-Clients

```
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());
    }
```

Abbildung 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() };
        }
```

Abbildung 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);
  }
}
```

Abbildung 28. AcceptAllX509TrustManager.java

JAAS-Konfiguration für Telemetriekanal

Konfigurieren Sie JAAS für die Authentifizierung des vom Client gesendeten Benutzernamens .

Der WebSphere MQ-Administrator konfiguriert unter Verwendung von JAAS, welche MQTT-Kanäle eine Clientauthentifizierung erfordern. Geben Sie den Namen einer JAAS-Konfiguration für jeden Kanal an, der die JAAS-Authentifizierung ausführen soll. Kanäle können alle dieselbe JAAS-Konfiguration verwenden, oder sie können verschiedene JAAS-Konfigurationen verwenden. Die Konfigurationen sind in *WMQData directory*\qmgrs*qMgrName*\mqxr\jaas.configdefiniert.

Die Datei `jaas.config` ist nach dem JAAS-Konfigurationsnamen organisiert. Unter jedem Konfigurationsnamen befindet sich eine Liste mit Anmeldekonfigurationen; siehe Abbildung 29 auf Seite 145.

JAAS stellt vier Standardanmeldemodule zur Verfügung. Die NT- und UNIX-Standardanmeldemodule sind begrenzt.

JndiLoginModule
Die Authentifizierung erfolgt auf Basis eines Verzeichnisservice, der in der Java Naming and Directory Interface (JNDI) konfiguriert ist.

Krb5LoginModule
Authentifiziert die Verwendung von Kerberos-Protokollen.

NTLoginModule
Authentifiziert unter Verwendung der NT-Sicherheitsinformationen für den aktuellen Benutzer.

UnixLoginModule
Die Authentifizierung erfolgt über die UNIX-Sicherheitsinformationen für den aktuellen Benutzer.

Bei `NTLoginModule` und `UnixLoginModule` besteht das Problem, dass der Telemetrieservice (MQXR) nicht mit der Identität des MQTT-Kanals, sondern mit der Identität `mqm` ausgeführt wird. `mqm` ist die Identität, die für die Authentifizierung an `NTLoginModule` oder `UnixLoginModule` übergeben wird, und nicht die Identität des Clients.

Um dieses Problem zu beheben, müssen Sie Ihr eigenes Anmeldemodul schreiben oder die anderen Standardanmeldemodule verwenden. Mit WebSphere MQ Telemetry wird das Musteranmeldemodul `JAASLoginModule.java` ausgeliefert. Es handelt sich um eine Implementierung der `javax.security.auth.spi.LoginModule` -Schnittstelle. Verwenden Sie diese Option, um Ihre eigene Authentifizierungsmethode zu entwickeln.

Alle neuen LoginModule-Klassen, die Sie bereitstellen, müssen sich auf dem Klassenpfad des Telemetrieservice (MQXR) befinden. Stellen Sie Ihre Klassen nicht in WebSphere MQ-Verzeichnisse, die sich im

Klassenpfad befinden. Erstellen Sie Ihre eigenen Verzeichnisse und definieren Sie den gesamten Klassenpfad für den Telemetrieservice (MQXR).

Sie können den Klassenpfad, der vom Telemetrieservice (MQXR) verwendet wird, erweitern, indem Sie den Klassenpfad in der Datei `service.env` definieren. `CLASSPATH` muss aktiviert sein, und die Klassenpfadanweisung kann nur Literale enthalten. Sie können keine Variablen in CLASSPATH verwenden, z. B. `CLASSPATH=%CLASSPATH%` ist nicht korrekt. Der Telemetrieservice (MQXR) legt seinen eigenen Klassenpfad fest. Der in `service.env` definierte CLASSPATH wird hinzugefügt.

Der Telemetrieservice (MQXR) stellt zwei Callbacks zur Verfügung, die Werte für `Benutzername` und `Kennwort` für einen Client liefern, der mit dem MQTT-Kanal verbunden ist. Der `Benutzername` und das `Kennwort` werden im Objekt `MqttConnectOptions` festgelegt. Im Abschnitt Abbildung 30 auf Seite 145 finden Sie ein Beispiel dafür, wie auf den `Benutzernamen` und das `Kennwort` zugegriffen wird.

Beispiele

Ein Beispiel für eine JAAS-Konfigurationsdatei mit einer benannten Konfiguration, `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";
};
```

Abbildung 29. Beispieldatei `jaas.config`

Dies ist ein Beispiel eines JAAS-Anmeldemoduls, das für den Empfang der Angaben von `Benutzername` und `Kennwort`, die von einem MQTT-Client zur Verfügung gestellt werden, codiert ist.

```
  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;
  }
```

Abbildung 30. `JAASLoginModule.Login()` -Beispielmethode

Konzepte des IBM WebSphere MQ Telemetry-Dämons für Geräte

Der IBM WebSphere MQ Telemetry-Dämon für Geräte ist eine erweiterte MQTT V3 -Clientanwendung. Damit können Sie Nachrichten von anderen MQTT-Clients speichern und weiterleiten (Store-and-forward-Verfahren). Der Dämon stellt wie ein MQTT-Client eine Verbindung zu IBM WebSphere MQ her, Sie können jedoch auch andere MQTT-Clients mit ihm verbinden.

Der Dämon ist ein Publish/Subscribe-Broker. MQTT V3-Clients verbinden sich mit ihm, um Themen zu veröffentlichen oder zu subskribieren. Dabei werden Themenzeichenfolgen für Veröffentlichungen und Themenfilter für Subskriptionen verwendet. Die Themenzeichenfolge ist hierarchisch aufgebaut und die einzelnen Themenebenen sind durch einen Schrägstrich (/) voneinander getrennt. Themenfilter sind Themenzeichenfolgen, die Platzhalter des Typs + bei einer einzelnen Ebene und den Platzhalter # bei mehreren Ebenen als letzten Teil der Themenzeichenfolge enthalten können.

Anmerkung: Die Platzhalter im Dämon unterliegen den restriktiveren Regeln von WebSphere Message Broker Version 6. In IBM WebSphere MQ ist dies anders. Dieses Produkt unterstützt mehrere Platzhalter für mehrere Ebenen. Platzhalter können für eine beliebige Anzahl an Ebenen in der Hierarchie stehen und sich an einer beliebigen Stelle in der Themenzeichenfolge befinden.

Mehrere MQTT v3-Clients verbinden sich unter Verwendung eines Empfangsprogrammports mit dem Dämon. Der standardmäßige Empfangsprogrammport ist änderbar. Sie können mehrere Empfangsprogrammports definieren und diesen unterschiedliche Namensbereiche zuordnen. Weitere Informationen hierzu finden Sie unter „Empfangsprogrammports für den WebSphere MQ Telemetry-Dämon für Geräte” auf Seite 154. Der Dämon ist selbst ein MQTT v3-Client. Konfigurieren Sie eine Dämonbridgeverbindung, um den Dämon mit dem Empfangsprogrammport eines anderen Dämons oder mit einem WebSphere MQ Telemetry-Service (MQXR) zu verbinden.

Sie können mehrere Bridges für den WebSphere MQ Telemetry-Dämon für Geräte konfigurieren. Mithilfe der Bridges können Sie zum Austausch von Veröffentlichungen ein Netz aus Dämonen miteinander verbinden.

Jede Bridge kann Themen bei ihrem lokalen Dämon veröffentlichen und subskribieren. Sie kann außerdem Themen bei einem anderen Dämon, einem WebSphere MQ-Publish/Subscribe-Broker oder einem beliebigen anderen MQTT v3-Broker, mit dem sie verbunden ist, veröffentlichen und subskribieren. Mithilfe eines Themenfilters können Sie die Veröffentlichungen auswählen, die von einem Broker an einen anderen weitergegeben werden sollen. Sie können Veröffentlichungen in beide Richtungen weitergeben. Sie können Veröffentlichungen aus dem lokalen Dämon an jeden seiner angehängten fernen Broker bzw. aus den angehängten Brokern an den lokalen Dämon weitergeben; siehe „Bridges des IBM WebSphere MQ Telemetry-Dämons für Geräte” auf Seite 146.

Bridges des IBM WebSphere MQ Telemetry-Dämons für Geräte

Eine Bridge des IBM WebSphere MQ Telemetry-Dämons für Geräte verbindet zwei Publish/Subscribe-Broker mithilfe des MQTT v3-Protokolls. Die Bridge gibt Veröffentlichungen von einem Broker zum anderen in beide Richtungen weiter. Am einen Ende befindet sich eine Bridgeverbindung des WebSphere MQ Telemetry-Dämons für Geräte. Am anderen Ende befindet sich ein Warteschlangenmanager oder ein anderer Dämon. Ein Warteschlangenmanager wird über einen Telemetriekanal mit der Bridgeverbindung verbunden. Ein Dämon wird über ein Dämonempfangsprogramm mit der Bridgeverbindung verbunden.

Der IBM WebSphere MQ Telemetry-Dämon für Geräte unterstützt eine oder mehrere gleichzeitige Verbindungen mit anderen Brokern. Die Verbindungen vom Dämon werden als Bridge bezeichnet und durch Verbindungseinträge in der Dämonkonfigurationsdatei definiert. Die Verbindungen zu IBM WebSphere MQ werden mithilfe von IBM WebSphere MQ Telemetry-Kanälen hergestellt, wie in der folgenden Abbildung dargestellt:

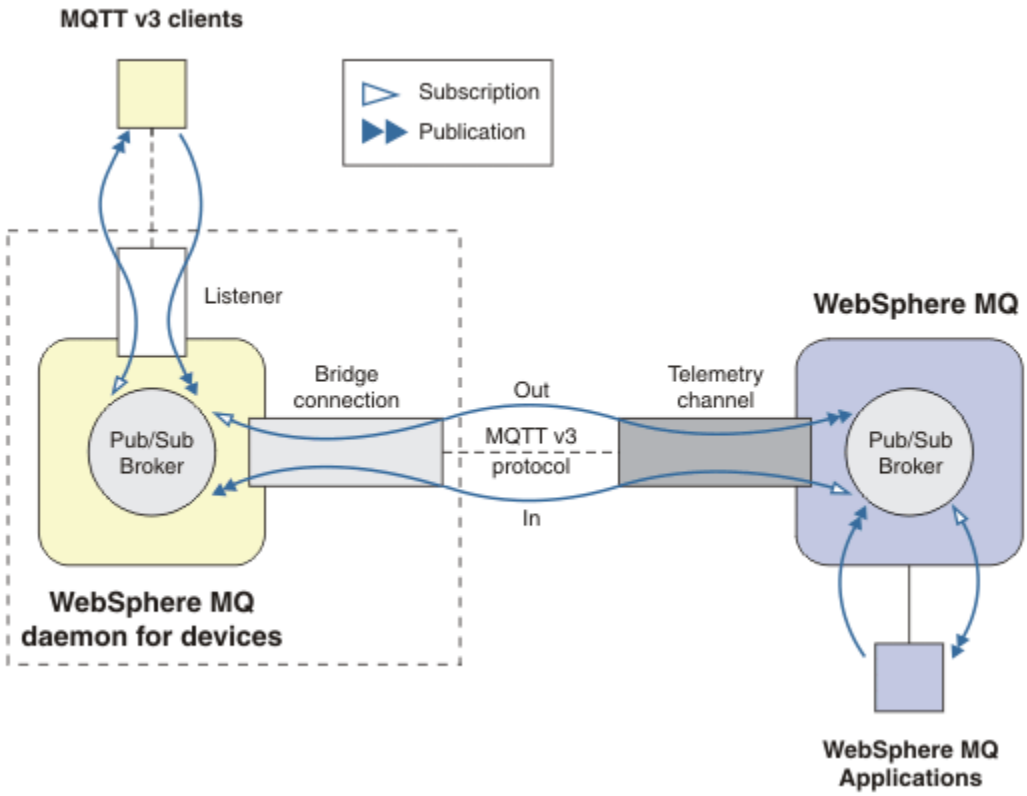


Abbildung 31. Verbindet IBM WebSphere MQ Telemetry daemon for devices mit IBM WebSphere MQ

Eine Bridge verbindet den Dämon mit einem anderen Broker in Form eines MQTT v3-Clients. Die Bridge-parameter spiegeln die Attribute eines MQTT v3-Clients.

Eine Bridge ist mehr als eine Verbindung. Sie dient als Veröffentlichungs- und Subskriptionsagent zwischen zwei Publish/Subscribe-Brokern. Der lokale Broker ist der IBM WebSphere MQ Telemetry-Dämon für Geräte, und der ferne Broker ist ein beliebiger Publish/Subscribe-Broker, der das MQTT v3-Protokoll unterstützt. Der ferne Broker ist in der Regel ein anderer Dämon oder IBM WebSphere MQ.

Die Bridge hat die Aufgabe, Veröffentlichungen zwischen den beiden Brokern weiterzugeben. Die Bridge ist bidirektional. Sie gibt Veröffentlichungen in beide Richtungen weiter. In Abbildung 31 auf Seite 147 ist dargestellt, wie die Bridge den IBM WebSphere MQ Telemetry-Dämon für Geräte mit IBM WebSphere MQ verbindet. Im Abschnitt „Beispiel für "topic"-Einstellungen für die Bridge” auf Seite 148 wird anhand von Beispielen dargestellt, wie die Bridge mithilfe des Parameters "topic" konfiguriert wird.

Die Pfeile In und Out in Abbildung 31 auf Seite 147 geben die Bidirektionalität der Bridge an. Am einen Ende des Pfeils wird eine Subskription erstellt. Die Veröffentlichungen, die der Subskription entsprechen, werden an den Broker am gegenüberliegenden Ende des Pfeils veröffentlicht. Der Pfeil wird gemäß der Flussrichtung von Veröffentlichungen gekennzeichnet. Veröffentlichungen fließen zum Dämon hin (In) und vom Dämon weg (Out). Die Kennzeichnungen sind wichtig, da sie in der Befehlssyntax verwendet werden. Mit In und Out wird die Flussrichtung von Veröffentlichungen angegeben und nicht der Ort, an den Subskriptionen gesendet werden.

Andere Clients, Anwendungen oder Broker können mit IBM WebSphere MQ oder mit dem WebSphere MQ Telemetry-Dämon für Geräte verbunden werden. Sie veröffentlichen und subskribieren Themen bei dem Broker, mit dem sie verbunden sind. Wenn es sich beim Broker um IBM WebSphere MQ handelt, können die Themen in Gruppen zusammengefasst oder verteilt sein. Zudem werden die Themen möglicherweise beim lokalen Warteschlangenmanager nicht explizit definiert.

Verwendung von Bridges

Verbinden Sie Dämonen mithilfe von Bridgeverbindungen und Empfangsprogrammen miteinander. Verbinden Sie Dämonen und Warteschlangenmanager mithilfe von Bridgeverbindungen und Telemetriekanälen. Wenn Sie mehrere Broker miteinander verbinden, entstehen möglicherweise Schleifen. Seien Sie vorsichtig: Es kann vorkommen, dass Veröffentlichungen unerkannt in einer Schleife von Brokern endlos zirkulieren.

Hier einige der Gründe für die Verwendung von Dämonen, die über eine Bridge mit IBM WebSphere MQ verbunden sind:

Geringere Anzahl von MQTT-Clientverbindungen mit WebSphere MQ
Wenn Sie eine Hierarchie aus Dämonen verwenden, können Sie viele Clients mit WebSphere MQ verbinden; mehr Clients als die Anzahl von Clients, die ein einzelner Warteschlangenmanager jeweils verbinden kann.

Speichern und Weiterleiten von Nachrichten zwischen MQTT-Clients und WebSphere MQ
Sie können das Verfahren zum Speichern und Weiterleiten verwenden, um zu vermeiden, dass zwischen Clients und IBM WebSphere MQ Dauerverbindungen bestehen, wenn die Clients keinen eigenen Speicher haben. Sie können mehrere Arten von Verbindungen zwischen dem MQTT-Client und WebSphere MQ verwenden. Informationen hierzu finden Sie unter Telemetry-Konzepte und Szenarios für die Überwachung und Steuerung.

Filtern der zwischen MQTT-Clients und WebSphere MQ ausgetauschten Veröffentlichungen
Veröffentlichungen werden üblicherweise in Nachrichten, die lokal verarbeitet werden, und Nachrichten, die andere Anwendungen einbeziehen, unterteilt. Lokale Veröffentlichungen enthalten Steuerungsabläufe zwischen Sensoren und Aktuatoren, während ferne Veröffentlichungen Anforderungen für Lesevorgänge, Statusinformationen und Konfigurationsbefehle enthalten.

Ändern des Themenbereichs von Veröffentlichungen
Verhindern Sie, dass Themenzeichenfolgen von Clients, die mit unterschiedlichen Empfangsprogrammports verbunden sind, miteinander in Konflikt geraten. Im Beispiel wird der Dämon verwendet, um Messungen von unterschiedlichen Gebäuden zu kennzeichnen. Informationen hierzu finden Sie unter Themenbereiche von unterschiedlichen Clientgruppen trennen.

Beispiel für "topic"-Einstellungen für die Bridge

Alles beim fernen Broker veröffentlichen - Standardwerte verwenden

Die Standardrichtung wird als out bezeichnet, und die Bridge veröffentlicht Themen beim fernen Broker. Der Parameter topic bestimmt mithilfe von Themenfiltern, welche Themen weitergegeben werden.

Die Bridge verwendet den Parameter topic in Abbildung 32 auf Seite 148, um all das zu subskribieren, was beim lokalen Dämon von MQTT-Clients oder von anderen Brokern veröffentlicht wird. Die Bridge veröffentlicht die Themen bei dem fernen Broker, der mit der Bridge verbunden ist.

```
connection Daemon1
topic #
```

Abbildung 32. Alles beim fernen Broker veröffentlichen

Alles beim fernen Broker veröffentlichen - explizit

Die Einstellung topic im folgenden Codefragment liefert dasselbe Ergebnis wie die Standardeinstellungen. Der einzige Unterschied besteht darin, dass der Parameter **direction** explizit ist. Verwenden Sie die Richtung `out`, um den lokalen Broker, also den Dämon, zu subskribieren und beim fernen Broker zu veröffentlichen. Auf dem lokalen Dämon erstellte Veröffentlichungen, die die Bridge subskribiert, werden beim fernen Broker veröffentlicht.

```
connection Daemon1
topic # out
```

Abbildung 33. Alles beim fernen Broker veröffentlichen - explizit

Alles beim lokalen Broker veröffentlichen

Anstelle der Richtung `out` können Sie die entgegengesetzte Richtung `in` festlegen. Im folgenden Codefragment wurde die Bridge so konfiguriert, dass sie alles subskribiert, was bei dem mit der Bridge verbundenen Broker veröffentlicht wird. Die Bridge veröffentlicht die Themen beim lokalen Broker, dem Dämon.

```
connection Daemon1
topic # in
```

Abbildung 34. Alles beim lokalen Broker veröffentlichen

Alles aus dem Exportthema beim lokalen Broker im Importthema beim fernen Broker veröffentlichen

Verwenden Sie die beiden zusätzlichen "topic"-Parameter **local_prefix** und **remote_prefix**, um den Themenfilter `#` in den vorherigen Beispielen zu ändern. Ein Parameter wird verwendet, um den in der Subskription verwendeten Themenfilter zu ändern, der andere, um das Thema, in dem die Veröffentlichung veröffentlicht wird, zu ändern. Dabei geht es darum, den Anfang der im einen Broker verwendeten Themenzeichenfolge durch eine andere Themenzeichenfolge im anderen Broker zu ersetzen.

Je nach der Richtung des Themenbefehls wird die Bedeutung von **local_prefix** und **remote_prefix** ins Gegenteil verkehrt. Wenn die Richtung `out` lautet, also die Standardeinstellung verwendet wird, wird **local_prefix** als Teil der Themensubskription verwendet und **remote_prefix** ersetzt den **local_prefix**-Teil der Themenzeichenfolge in der fernen Veröffentlichung. Wenn die Richtung `in` lautet, wird **remote_prefix** Teil der fernen Subskription und **local_prefix** ersetzt den **remote_prefix**-Teil der Themenzeichenfolge.

Der erste Teil einer Themenzeichenfolge dient häufig zur Definition eines Themenbereichs. Verwenden Sie die zusätzlichen Parameter, um den Themenbereich zu ändern, in dem ein Thema veröffentlicht wird. Sie können dies tun, um zu verhindern, dass das weitergegebene Thema mit einem anderen Thema des Zielbrokers in Konflikt gerät, oder um die Themenzeichenfolge eines Mountpunkts zu entfernen.

Beispiel: Im folgenden Codefragment werden alle Veröffentlichungen in der Themenzeichenfolge `export/#` beim Dämon in `import/#` beim fernen Broker erneut veröffentlicht.

```
topic # out export/ import/
```

Abbildung 35. Alles aus dem Exportthema beim lokalen Broker im Importthema beim fernen Broker veröffentlichen

Alles im Importthema beim lokalen Broker aus dem Exportthema beim fernen Broker veröffentlichen

Im folgenden Codefragment ist die Konfiguration umgekehrt dargestellt. Die Bridge subskribiert alles, was mit der Themenzeichenfolge `export/#` beim fernen Broker veröffentlicht wird, und veröffentlicht es in der Themenzeichenfolge `import/#` beim lokalen Broker.

```
connection Daemon1
topic # in import/ export/
```

Abbildung 36. Alles im Importthema beim lokalen Broker aus dem Exportthema beim fernen Broker veröffentlichen

Mit den ursprünglichen Themenzeichenfolgen alles aus dem Mountpunkt 1884/ beim fernen Broker veröffentlichen

Im folgenden Codefragment subskribiert die Bridge alles, was von Clients veröffentlicht wird, die mit dem Mountpunkt 1884/ am lokalen Dämon verbunden sind. Die Bridge veröffentlicht alles, was am Mountpunkt des fernen Brokers veröffentlicht wird. Die Zeichenfolge 1884/ des Mountpunkts wird aus den Themen entfernt, die am fernen Broker veröffentlicht werden. Der Parameter *local_prefix* ist mit der Zeichenfolge 1884/ des Mountpunkts identisch. Beim Parameter *remote_prefix* handelt es sich um eine leere Zeichenfolge.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Abbildung 37. Mit den ursprünglichen Themenzeichenfolgen alles aus dem Mountpunkt 1884/ beim fernen Broker veröffentlichen

Themenbereiche von unterschiedlichen Clients trennen, die mit unterschiedlichen Dämonen verbunden sind

Für Stromzähler wird eine Anwendung geschrieben, um Zählerstände für ein Gebäude zu veröffentlichen. Die Zählerstände werden mithilfe von MQTT-Clients bei einem Dämon veröffentlicht, der in demselben Gebäude betrieben wird. Für die Veröffentlichungen wird das Thema `power` ausgewählt. Die Anwendung wird in mehreren Gebäuden in einem Komplex implementiert. Zur Standortüberwachung und Datenspeicherung werden Zählerstände von allen Gebäuden mithilfe von Bridgeverbindungen zusammengefasst. Mithilfe der Verbindungen werden alle Dämonen in demselben Gebäude an einem zentralen Standort mit WebSphere MQ verknüpft.

Die Clientanwendungen in jedem Gebäude sind identisch, aber die Daten müssen nach Gebäude unterschieden werden. Jede Ablesung hat ein Thema `power` und muss mit der Gebäudenummer als Präfix versehen werden, um sie zu unterscheiden. Die Bridge vom ersten Gebäude im Komplex weist das Präfix `meters/building01/` auf, die vom zweiten Gebäude das Präfix `meters/building02/`. Die Zählerstände der anderen Gebäude folgen diesem Muster. WebSphere MQ empfängt die gemessenen Werte mit Themen wie `meters/building01/power`.

Das Beispiel ist ein Beispiel. In der Praxis ist der Topicbereich, in dem die Anwendung veröffentlicht, wahrscheinlich konfigurierbar.

Die Konfigurationsdatei für die einzelnen Dämonen enthält eine Themenanweisung, die dem Muster im folgenden Codefragment folgt:

```
connection Daemon1
topic power out "" meters/building01/
```

Abbildung 38. Themenbereiche von Clients trennen, die mit unterschiedlichen Dämonen verbunden sind

Geben Sie eine leere Zeichenfolge als Platzhalter für den nicht verwendeten Parameter local_prefix an.

Themenbereiche von Clients trennen, die mit einem Dämon verbunden sind

Nehmen wir an, dass nur ein Dämon verwendet wird, um alle Stromzähler zu verbinden. Vorausgesetzt, die Anwendung kann für die Verbindung mit unterschiedlichen Ports konfiguriert werden, können Sie die Gebäude unterscheiden, indem Sie die Zähler von unterschiedlichen Gebäuden mit unterschiedlichen Empfangsprogrammports verbinden. Dies wird im folgenden Codefragment dargestellt. Auch dieses Beispiel ist erfunden. Es zeigt, wie Mountpunkte verwendet werden können.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Abbildung 39. Themenbereiche von Clients trennen, die mit einem Dämon verbunden sind

Unterschiedliche Themen für Veröffentlichungen neu zuordnen, die in beide Richtungen fließen

In der Konfiguration im folgenden Codefragment subskribiert die Bridge das einzelne Thema b beim fernen Broker und leitet Veröffentlichungen zu b an den lokalen Dämon weiter, wobei das Thema in ageändert wird. Die Bridge subskribiert auch das einzelne Thema x beim lokalen Broker und leitet Veröffentlichungen zu x an den fernen Broker weiter, wobei das Thema in ygeändert wird.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Abbildung 40. Unterschiedliche Themen für Veröffentlichungen neu zuordnen, die in beide Richtungen fließen

Ein wichtiger Aspekt bei diesem Beispiel ist, dass unterschiedliche Themen bei beiden Brokern subskribiert und veröffentlicht werden. Die Themenbereiche der beiden Broker überschneiden sich nicht.

Dieselben Themen für Veröffentlichungen neu zuordnen, die in beide Richtungen fließen (Schleife)

Im Gegensatz zum vorherigen Beispiel ergibt die Konfiguration in Abbildung 41 auf Seite 152 generell eine Schleife. In der Themenanweisung `topic "" in a b` subskribiert die Bridge b über Fernzugriff und veröffentlicht Daten in a lokal. In der anderen Themenanweisung subskribiert die Bridge a lokal und veröffentlicht Daten in b über Fernzugriff. Diese Konfiguration kann auch wie in Abbildung 42 auf Seite 152 dargestellt geschrieben werden.

Das allgemeine Ergebnis ist, dass die Veröffentlichung als Veröffentlichung zum Thema aan den lokalen Dämon übertragen wird, wenn ein Client fern in b veröffentlicht. Bei der Veröffentlichung durch die Brücke zum lokalen Dämon für das Thema astimmt die Veröffentlichung jedoch mit der Subskription überein, die durch die Brücke zum lokalen Thema aerstellt wurde. Die Subskription lautet `topic "" out a b`. Infolgedessen wird die Veröffentlichung als Veröffentlichung zum Thema bzurück an den fernen Broker übertragen. Die Bridge ist jetzt für das ferne Thema bsubskribiert und der Zyklus beginnt erneut.

Einige Broker implementieren eine Funktion zum Auffinden von Schleifen, um Schleifen zu verhindern. Die Funktion zum Auffinden von Schleifen muss jedoch funktionieren, wenn unterschiedliche Arten von Brokern über Bridges miteinander verbunden sind. Die Funktion zum Auffinden von Schleifen funktioniert nicht, wenn WebSphere MQ über eine Bridge mit dem WebSphere MQ Telemetry-Dämon für Geräte verbunden ist. Sie funktioniert, wenn zwei IBM WebSphere MQ Telemetry-Dämonen für Geräte über eine Bridge miteinander verbunden sind. Die Funktion zum Auffinden von Schleifen ist standardmäßig aktiviert. Informationen hierzu finden Sie unter try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

Abbildung 41. !Ordnen Sie dieselben Themen für Veröffentlichungen zu, die in beide Richtungen fließen

```
connection Daemon1
topic "" both a b
```

Abbildung 42. !Ordnen Sie dieselben Themen für Veröffentlichungen, die in beide Richtungen fließen, mithilfe von `both`neu zu.

Die Konfigurationen in Abbildung 40 auf Seite 151 und in Abbildung 41 auf Seite 152 sind identisch.

Verfügbarkeit von Bridgeverbindungen für den IBM WebSphere MQ Telemetry daemon for devices

Konfigurieren Sie mehrere Adressen von Bridgeverbindungen für den IBM WebSphere MQ Telemetry daemon for devices, um eine Verbindung mit dem ersten verfügbaren fernen Broker herzustellen. Wenn es sich beim Broker um einen Mehrinstanz-Warteschlangenmanager handelt, geben Sie beide TCP/IP-Adressen an. Konfigurieren Sie eine primäre Verbindung zum Herstellen oder Wiederherstellen einer Verbindung mit dem primären Server, sofern er verfügbar ist.

Beim Verbindungsbridgeparameter addresses handelt es sich um eine Liste mit TCP/IP-Socketadressen. Die Bridge versucht, der Reihe nach eine Verbindung mit den einzelnen Adressen herzustellen, bis mit einer Adresse eine erfolgreiche Verbindung hergestellt werden kann. Mit den Verbindungsparametern round_robin und start_type wird festgelegt, wie die Adressen nach dem erfolgreichen Herstellen einer Verbindung verwendet werden.

Wenn start_type auf auto, manual oder lazy festgelegt wird und die Verbindung fehlschlägt, versucht die Bridge, die Verbindung wiederherzustellen. Sie verwendet die einzelnen Adressen der Reihe nach mit einer Verzögerung von etwa 20 Sekunden zwischen den einzelnen Verbindungsversuchen. Wenn `start_type` auf once festgelegt wird und die Verbindung fehlschlägt, versucht die Bridge nicht automatisch, die Verbindung wiederherzustellen.

Wenn round_robin auf true gesetzt ist, versucht die Bridgeverbindung bei der ersten Adresse in der Liste zu starten und probiert die einzelnen Adressen in der Liste der Reihe nach aus. Wenn sie am Ende der Liste anlangt, beginnt sie wieder bei der ersten Adresse. Wenn die Liste nur eine Adresse enthält, wiederholt sie den Versuch alle 20 Sekunden.

Wenn round_robin auf false gesetzt ist, hat die erste Adresse in der Liste, die als "primärer Server" bezeichnet wird, Vorrang. Wenn der erste Versuch, mit dem primären Server eine Verbindung herzustellen, fehlschlägt, versucht die Bridge wiederholt im Hintergrund, zu dem Server eine Verbindung herzustellen. Gleichzeitig versucht die Bridge, eine Verbindung mithilfe der anderen Adressen in der Liste herzustellen. Wenn die Versuche im Hintergrund, zum primären Server eine Verbindung herzustellen, erfolgreich sind, trennt die Bridge die aktuelle Verbindung und wechselt zur Verbindung mit dem primären Server.

Wenn eine Verbindung beispielsweise durch Ausgabe des Befehls **connection_stop** freiwillig getrennt wird und dann versucht wird, die Verbindung wiederherzustellen, wird wieder dieselbe Adresse verwendet. Wenn die Verbindung aufgrund eines Verbindungsfehlers getrennt wird oder weil der ferne Broker die Verbindung trennt, wartet die Bridge 20 Sekunden. Dann versucht sie, mit der nächsten Adresse in der Liste oder, wenn die Liste nur eine Adresse enthält, mit derselben Adresse eine Verbindung herzustellen.

Verbindung mit einem Mehrinstanz-Warteschlangenmanager herstellen

In der Konfiguration eines Mehrinstanz-Warteschlangenmanagers wird der Warteschlangenmanager auf zwei verschiedenen Servern mit unterschiedlichen IP-Adressen ausgeführt. Telemetriekanäle werden in

der Regel ohne bestimmte IP-Adresse konfiguriert. Sie werden lediglich mit einer Portnummer konfiguriert. Der Telemetriekanal wählt beim Start standardmäßig die erste verfügbare Netzadresse auf dem lokalen Server aus.

Konfigurieren Sie den Parameter addresses der Bridgeverbindung mit den beiden vom Warteschlangenmanager verwendeten IP-Adressen. Setzen Sie round_robin auf "true".

Wenn die aktive Warteschlangenmanagerinstanz ausfällt, schaltet der Warteschlangenmanager zur Standby-Instanz um. Der Dämon erkennt, dass die Verbindung mit der aktiven Instanz getrennt wurde, und versucht, die Verbindung mit der Standby-Instanz wiederherzustellen. Er verwendet die andere IP-Adresse in der für die Bridgeverbindung konfigurierten Adressliste.

Der Warteschlangenmanager, mit dem die Bridge eine Verbindung herstellt, ist immer noch derselbe Warteschlangenmanager. Der Warteschlangenmanager stellt seinen eigenen Status wieder her. Wenn cleansession auf "false" gesetzt ist, wird die Sitzung der Bridgeverbindung im Status vor der Übernahme wiederhergestellt. Die Verbindung wird nach einer Verzögerung wieder aufgenommen. Nachrichten mit der Servicequalität "mindestens einmal" oder "höchstens einmal" gehen nicht verloren und Subskriptionen funktionieren weiterhin.

Die für die Verbindungswiederholung erforderliche Zeit hängt zum einen von der Anzahl der Kanäle und Clients ab, die beim Start der Standby-Instanz erneut starten, und zum anderen von der Anzahl der unvollständig verarbeiteten Nachrichten. Die Bridgeverbindung versucht möglicherweise, wiederholt eine Verbindung mit beiden IP-Adressen herzustellen, bevor die Verbindung wiederhergestellt werden kann.

Konfigurieren Sie den Telemetriekanal eines Mehrinstanz-Warteschlangenmanagers nicht mit einer bestimmten IP-Adresse. Die IP-Adresse ist nur auf einem Server gültig.

Wenn Sie eine andere Lösung für hohe Verfügbarkeit verwenden, mit der die IP-Adresse verwaltet wird, ist es möglicherweise in Ordnung, einen Telemetriekanal mit einer bestimmten IP-Adresse zu konfigurieren.

cleansession

Eine Bridgeverbindung ist eine MQTT v3-Clientsitzung. Sie können festlegen, ob eine Verbindung eine neue Sitzung startet oder eine bestehende Sitzung wiederherstellt. Wenn sie eine bestehende Sitzung wiederherstellt, behält die Bridgeverbindung die Subskriptionen und ständigen Veröffentlichungen aus der vorherigen Sitzung bei.

Legen Sie cleansession nicht auf "false" fest, wenn addresses mehrere IP-Adressen enthält und die IP-Adressen eine Verbindung mit Telemetriekanälen, die von unterschiedlichen Warteschlangenmanagern gehostet werden, oder mit unterschiedlichen Telemetriedämonen herstellen. Der Sitzungsstatus wird zwischen Warteschlangenmanagern oder Dämonen nicht übertragen. Beim Versuch, eine bestehende Sitzung auf einem anderen Warteschlangenmanager oder Dämon erneut zu starten, wird eine neue Sitzung gestartet. Unbestätigte Nachrichten gehen verloren, und Subskriptionen verhalten sich möglicherweise nicht erwartungsgemäß.

Benachrichtigungen

Eine Anwendung kann mithilfe von Benachrichtigungen verfolgen, ob die Bridgeverbindung aktiv ist. Bei einer Benachrichtigung handelt es sich um eine Veröffentlichung mit dem Wert 1 für eine aktive Verbindung und dem Wert 0 für eine getrennte Verbindung. Er wird in *topicString* veröffentlicht, definiert durch den Parameter `notification_topic` . Der Standardwert von *topicString* ist $SYS/`broker`/`connection`/ *clientIdentifier*/`state`. Die Standardzeichenfolge *topicString* enthält das Präfix $SYS. Subskribieren Sie Themen, die mit $SYS beginnen, indem Sie einen Themenfilter definieren, der mit $SYS beginnt. Der Themenfilter # (alles subskribieren) subskribiert keine Themen, die auf dem Dämon mit $SYS beginnen. Betrachten Sie $SYS als Definition eines speziellen System-Topic-Bereichs, der sich vom Anwendungs-Topic-Bereich unterscheidet.

Benachrichtigungen ermöglichen es IBM WebSphere MQ Telemetry daemon for devices , MQTT-Clients zu benachrichtigen, wenn eine Bridge verbunden oder getrennt wird.

keepalive_interval

Der Parameter keepalive_interval für Bridgeverbindungen legt das Intervall zwischen TCP/IP-Ping-Befehlen von der Bridge an den fernen Server fest. Das Standardintervall beträgt 60 Sekunden. Die Überprüfung mit Ping verhindert, dass die TCP/IP-Sitzung vom fernen Server oder von einer Firewall beendet wird, die für die Verbindung einen Inaktivitätszeitraum erkennt.

clientId

Eine Bridgeverbindung ist eine MQTT v3-Clientsitzung mit einem `clientIdentifier`, der durch den Parameter clientid der Bridgeverbindung festgelegt wird. Wenn Sie angeben, dass bei Verbindungswiederholungen eine vorherige Sitzung wiederaufgenommen wird, indem Sie den Parameter cleansession auf false setzen, muss in den einzelnen Sitzungen derselbe `clientIdentifier` verwendet werden. Der Standardwert von clientid lautet *hostname.connectionName* und bleibt derselbe.

Installation, Prüfung, Konfiguration und Steuerung des WebSphere MQ Telemetry-Dämons für Geräte

Die Installation, Konfiguration und Steuerung des Dämons erfolgt dateibasiert.

Installieren Sie den Dämon, indem Sie das Software Development Kit auf das Gerät kopieren, auf dem der Dämon ausgeführt werden soll.

Führen Sie beispielsweise das MQTT-Clientdienstprogramm aus und stellen Sie eine Verbindung zum WebSphere MQ Telemetry-Dämon für Geräte als Publish/Subscribe-Broker her. Weitere Informationen finden Sie unter Nachricht in einem bestimmten MQTT v3 -Client veröffentlichen .

Konfigurieren Sie den Dämon, indem Sie eine Konfigurationsdatei erstellen; siehe Konfigurationsdatei für WebSphere MQ Telemetry-Dämon für Geräte.

Sie können einen aktiven Dämon durch die Erstellung von Befehlen in der Datei `amqtdd.upd` steuern. Alle fünf Sekunden liest der Dämon die Datei, führt die Befehle aus und löscht die Datei; siehe Befehlsdatei für WebSphere MQ Telemetry-Dämon für Geräte.

Empfangsprogrammports für den WebSphere MQ Telemetry-Dämon für Geräte

Verbinden Sie MQTT V3-Clients unter Verwendung von Empfangsprogrammports mit dem WebSphere MQ Telemetry-Dämon für Geräte. Sie können einen Empfangsprogrammport durch einen Mountpunkt und eine maximale Anzahl an Verbindungen näher bestimmen.

Ein Empfangsprogrammport muss der Portnummer entsprechen, die in der MQTT-Clientmethode `connect(serverURI)` eines Clients angegeben ist, der mit diesem Port verbunden ist. Er hat sowohl auf dem Client als auch dem Dämon den Standardwert 1883.

Sie können den Standardport für den Dämon ändern, indem Sie die globale Definition für `Port` in der Dämonkonfigurationsdatei festlegen. Sie können bestimmte Ports festlegen, indem Sie der Dämonkonfigurationsdatei eine Definition für das `Empfangsprogramm` hinzufügen.

Sie können für jeden Empfangsprogrammport außer dem Standardport einen Mountpunkt für die Eingrenzung von Clients angeben. Clients, die mit einem Port verbunden sind, der über einen Mountpunkt verfügt, werden von anderen Clients isoliert; weitere Informationen finden Sie unter „Mountpunkte für den WebSphere MQ Telemetry-Dämon für Geräte” auf Seite 155.

Sie können die Anzahl der Client begrenzen, die eine Verbindung zu einem beliebigen Port herstellen können. Legen Sie die globale Definition `max_connections` fest, um Verbindungen mit dem Standardport zu begrenzen, oder bestimmen Sie jeden Empfangsprogrammport näher mit `max_connections`.

Beispiel

Es folgt das Beispiel einer Konfigurationsdatei, die den Standardport 1883 in 1880 ändert und Verbindungen mit dem Port 1880 in 10000 ändert. Verbindungen mit dem Port 1884 sind auf 1000 begrenzt.

Clients, die an den Port 1884 angehängt sind, werden von Clients isoliert, die an andere Port angehängt sind.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Mountpunkte für den WebSphere MQ Telemetry-Dämon für Geräte

Sie können einen Mountpunkt mit dem Port des Empfangsprogramms verbinden, der von MQTT-Clients verwendet wird, um eine Verbindung mit einem WebSphere MQ Telemetry-Dämon für Geräte herzustellen. Ein Mountpunkt grenzt die Veröffentlichungen und Subskriptionen ein, die von MQTT-Clients ausgetauscht werden, und verwendet hierzu einen Empfangsprogrammport von MQTT-Clients, die mit einem anderen Empfangsprogrammport verbunden sind.

Clients, die über einen Mountpunkt mit einem Empfangsprogrammport verbunden sind, können mit Clients, die mit einem anderen Empfangsprogrammport verbunden sind, keine Themen direkt austauschen. Clients, die ohne Mountpunkt mit einem Empfangsprogrammport verbunden sind, können auf beliebigen Clients Themen veröffentlichen oder subskribieren. Clients erkennen nicht, ob sie über einen Mountpunkt verbunden sind, und erstellen daher immer dieselben Themenzeichenfolgen.

Bei einem Mountpunkt handelt es sich um eine Textzeichenfolge, die der Themenzeichenfolge von Veröffentlichungen und Subskriptionen vorangestellt wird. Sie wird allen Themenzeichenfolgen vorangestellt, die von Clients erstellt werden, die über einen Mountpunkt mit einem Empfangsprogrammport verbunden sind. Die Textzeichenfolge wird aus allen Themenzeichenfolgen entfernt, die an Clients gesendet werden, die mit dem Empfangsprogrammport verbunden sind.

Wenn ein Empfangsprogrammport keinen Mountpunkt aufweist, werden die Themenzeichenfolgen von Veröffentlichungen und Subskriptionen nicht geändert, die von Clients erstellt und empfangen werden, die mit dem Port verbunden sind.

Erstellen Sie Mountpunktzeichenfolgen mit einem abschließenden `/`. Auf diese Weise wird der Mountpunkt zum übergeordneten Thema der Themenstruktur für den Mountpunkt.

Beispiel

Eine Konfigurationsdatei enthält folgende Empfangsprogrammports:

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

Ein mit Port 1883 verbundener Client erstellt eine Subskription für `MyTopic`. Der Dämon registriert die Subskription als `1883/MyTopic`. Ein anderer mit Port 1883 verbundener Client veröffentlicht eine Nachricht zum Thema `MyTopic`. Der Dämon ändert die Themenzeichenfolge in `1883/MyTopic` und sucht nach passenden Subskriptionen. Der Subskribent an Port 1883 empfängt die Veröffentlichung mit der ursprünglichen Themenzeichenfolge `MyTopic`. Der Dämon hat das Mountpunktpräfix aus der Themenzeichenfolge entfernt.

Ein anderer, mit Port 1884 verbundener Client veröffentlicht ebenfalls Daten zum Thema `MyTopic`. Dieses Mal registriert der Dämon das Thema als `1884/MyTopic`. Der Subskribent an Port 1883 erhält die Veröffentlichung nicht, da die unterschiedlichen Mountpunkte dazu führen, dass die Subskriptionen unterschiedliche Themenzeichenfolgen enthalten.

Ein mit Port 1885 verbundener Client veröffentlicht Daten zum Thema `1883/MyTopic`. Der Dämon ändert die Themenzeichenfolge nicht. Der Subskribent an Port 1883 empfängt die Veröffentlichung zu `MyTopic`.

Servicequalität, permanente Subskriptionen und ständige Veröffentlichungen für den WebSphere MQ Telemetry-Dämon für Geräte

Einstellungen für Servicequalität gelten nur für einen aktiven Dämon. Wenn ein Dämon kontrolliert oder durch einen Fehler beendet wird, geht der Status von unvollständigen Nachrichten verloren. Die Servicequalität "Mindestens einmal" oder "Höchstens einmal" für die Zustellung einer Nachricht kann nicht garantiert werden, wenn der Dämon beendet wird. Der WebSphere MQ Telemetry-Dämon für Geräte unterstützt eine begrenzte Persistenz. Legen Sie den Konfigurationsparameter **retained_persistence** fest, damit ständige Veröffentlichungen und Subskriptionen gespeichert werden, wenn der Dämon beendet wird.

Im Gegensatz zu WebSphere MQerfasst der WebSphere MQ Telemetry -Dämon für Geräte keine persistenten Daten. Sitzungsstatus, Nachrichtenstatus und ständige Veröffentlichungen werden nicht über Transaktionen gespeichert. Der Dämon löscht standardmäßig alle Daten, wenn er beendet wird. Sie können eine Option festlegen, um Subskriptionen und ständige Veröffentlichungen regelmäßig zu prüfen. Der Nachrichtenstatus geht immer verloren, wenn der Dämon beendet wird. Alle nicht ständigen Veröffentlichungen gehen verloren.

Setzen Sie die Konfigurationsoption `Retained_persistence` des Dämons auf `true`, um ständige Veröffentlichungen regelmäßig in einer Datei zu speichern. Beim Neustart des Dämons werden die zuletzt automatisch gespeicherten ständigen Veröffentlichungen wiederhergestellt. Von Clients erstellte ständige Nachrichten werden beim Neustart des Dämons standardmäßig nicht wiederhergestellt.

Setzen Sie die Konfigurationsoption `Retained_persistence` des Dämons auf `true`, um in einer persistenten Sitzung erstellte Subskriptionen regelmäßig in einer Datei zu speichern. Wenn `Retained_persistence` auf `true` festgelegt wird, werden Subskriptionen wiederhergestellt, die Clients in einer "persistenten Sitzung" erstellen, also in einer Sitzung, bei der `CleanSession` auf `false` festgelegt ist. Der Dämon stellt beim Neustart die Subskriptionen wieder her, die den Empfang von Veröffentlichungen starten. Der Client empfängt die Veröffentlichungen bei einem Neustart, bei dem `CleanSession` auf `false` festgelegt ist. Der Status der Clientsitzung wird beim Beenden eines Dämons standardmäßig nicht gespeichert. Somit werden Subskriptionen nicht wiederhergestellt, auch wenn der Client `CleanSession` auf `false` festlegt.

`Retained_persistence` ist eine Funktion zum automatischen Speichern. Sie speichert die letzten ständigen Veröffentlichungen oder Subskriptionen möglicherweise nicht. Sie können festlegen, wie häufig ständige Veröffentlichungen und Subskriptionen gespeichert werden. Legen Sie das Intervall zwischen Speichervorgängen oder die Anzahl der Änderungen zwischen Speichervorgängen fest. Verwenden Sie hierzu die Konfigurationsoptionen `autosave_on_changes` und `autosave_interval`.

Beispielkonfiguration zum Festlegen der Persistenz

```
# 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
```

Sicherheit des WebSphere MQ Telemetry-Dämons für Geräte

Der WebSphere MQ Telemetry-Dämon für Geräte kann Clients authentifizieren, die mit ihm verbunden sind, für die Verbindung mit anderen Broker Berechtigungsnachweise verwenden und den Zugriff auf Themen steuern. Die vom Dämon bereitgestellte Sicherheit ist begrenzt, da sie mithilfe des WebSphere MQ Telemetry C-Clients erstellt wird, der keinen SSL-Support bietet. Folglich werden Verbindungen mit dem Dämon nicht verschlüsselt und können nicht mithilfe von Zertifikaten authentifiziert werden.

Standardmäßig sind keine Sicherheitsfunktionen aktiviert.

Authentifizierung von Clients

MQTT-Clients können einen Benutzernamen und ein Kennwort mithilfe der Methoden `MqttConnectOptions.setUserName` und `MqttConnectOptions.setPassword` festlegen.

Ein Client, der eine Verbindung mit dem Dämon herstellt, wird authentifiziert, indem der von ihm angegebene Benutzername und das zugehörige Kennwort anhand von Einträgen in der Kennwortdatei überprüft werden. Um die Authentifizierung zu ermöglichen, erstellen Sie eine Kennwortdatei und legen Sie den Parameter `password_file` in der Dämonkonfigurationsdatei fest. Informationen hierzu finden Sie unter password_file.

Legen Sie den Parameter `allow_anonymous` in der Dämonkonfigurationsdatei fest, um Clients das Herstellen einer Verbindung mit einem Dämon, der die Authentifizierung überprüft, ohne Benutzername und Kennwort zu ermöglichen. Informationen hierzu finden Sie unter allow_anonymous. Wenn ein Client einen Benutzernamen oder ein Kennwort angibt, werden diese Angaben immer mit der Kennwortdatei abgeglichen, sofern der Parameter `password_file` festgelegt wurde.

Legen Sie den Parameter `clientid_prefixes` in der Dämonkonfigurationsdatei fest, um Verbindungen auf bestimmte Clients zu beschränken. Die Clients müssen `clientIdentifiers` aufweisen, die mit einem der im Parameter `clientid_prefixes` aufgeführten Präfixe beginnen. Informationen hierzu finden Sie unter clientid_prefixes.

Sicherheit für Bridgeverbindungen

Jede Bridgeverbindung des WebSphere MQ Telemetry-Dämons für Geräte entspricht einem MQTT V3-Client. Sie können den Benutzernamen und das Kennwort für jede Bridgeverbindung in Form eines Bridgeverbindungsparameters in der Dämonkonfigurationsdatei festlegen. Informationen hierzu finden Sie unter username und password. Danach kann eine Bridge sich einem Broker gegenüber selbst authentifizieren.

Zugriffssteuerung bei Themen

Für die Authentifizierung von Clients kann der Dämon auch den Zugriff auf Themen für die einzelnen Benutzer steuern. Der Dämon steuert den Zugriff, indem er das Thema, zu dem ein Client Daten veröffentlicht oder das ein Client subskribiert, mit einer Zugriffsthemenzeichenfolge in der Zugriffssteuerungsliste abgleicht. Informationen hierzu finden Sie unter acl_file.

Die Zugriffssteuerungsliste besteht aus zwei Teilen. Der erste Teil steuert den Zugriff für alle Clients, auch für anonyme Clients. Der zweite Teil enthält einen Abschnitt für alle Benutzer in der Kennwortdatei. Hier ist die Vergabe von Zugriffsrechten für die einzelnen Benutzer aufgeführt.

Beispiel

Die Sicherheitsparameter sind im folgenden Beispiel dargestellt.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Abbildung 43. Dämonkonfigurationsdatei

```
Fred:Fredpassword
Barney:Barneypassword
```

Abbildung 44. Kennwortdatei "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/#
```

Abbildung 45. Zugriffssteuerungsdatei "acl.txt"

Multicast verwalten

Verwenden Sie diese Informationen, um mehr über die WebSphere MQ Multicast-Verwaltungstasks wie das Verringern der Größe von Multicastnachrichten und das Aktivieren der Datenkonvertierung zu erfahren.

Erste Schritte mit Multicasting

Verwenden Sie diese Informationen als Einführung in WebSphere MQ Multicast-Themen und Kommunikationsinformationsobjekte.

Informationen zu diesem Vorgang

WebSphere MQ Multicast-Messaging verwendet das Netz, um Nachrichten durch Zuordnung von Themen zu Gruppenadressen zuzustellen. Mit den folgenden Tasks können Sie schnell testen, ob die erforderliche IP-Adresse und der erforderliche Port für Multicast-Messaging korrekt konfiguriert sind.

Erstellen eines COMMINFO-Objekts für Multicasting

Das Objekt Kommunikationsinformationen (COMMINFO) enthält die Attribute, die der Multicastübertragung zugeordnet sind. Weitere Informationen zu den Parametern des Befehls COMMINFO finden Sie in DEFINE COMMINFO .

Verwenden Sie das folgende Befehlszeilenbeispiel, um ein COMMINFO-Objekt für Multicasting zu definieren:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

Hierbei steht *MC1* für den Namen Ihres COMMINFO-Objekts, *Gruppenadresse* für die Multicast-IP-Adresse oder den DNS-Namen der Gruppe, und *Portnummer* für den Port für die Übertragung (Der Standardwert ist 1414).

Es wird ein neues COMMINFO-Objekt mit dem Namen *MC1* erstellt. Dieser Name ist der Name, den Sie beim Definieren eines TOPIC-Objekts im nächsten Beispiel angeben müssen.

TOPIC-Objekt für Multicasting erstellen

Ein Thema ist der Gegenstand der Informationen, die in einer Publish/Subscribe-Nachricht veröffentlicht werden, und ein Thema wird definiert, indem ein TOPIC-Objekt erstellt wird. TOPIC-Objekte verfügen über zwei Parameter, die definieren, ob sie mit Multicasting verwendet werden können. Diese Parameter sind: **COMMINFO** und **MCAST**.

- **COMMINFO** Dieser Parameter gibt den Namen des Multicastkommunikationsinformationsobjekts an. Weitere Informationen zu den Parametern des Befehls COMMINFO finden Sie in DEFINE COMMINFO .
- **MCAST** Dieser Parameter gibt an, ob Multicasting an dieser Position in der Themenstruktur zulässig ist.

Verwenden Sie das folgende Befehlszeilenbeispiel, um ein TOPIC-Objekt für Multicasting zu definieren:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

Es wird ein neues TOPIC-Objekt mit dem Namen *ALLSPORTS* erstellt. Sie hat eine Themenzeichenfolge *Sports*, das zugehörige Kommunikationsinformationsobjekt hat den Namen *MC1* (dies ist der Name, den Sie bei der Definition eines COMMINFO-Objekts im vorherigen Beispiel angegeben haben), und Multicasting ist aktiviert.

Testen des Multicast-Publish/Subscribe

Nachdem die Objekte "TOPIC" und "COMMINFO" erstellt wurden, können sie mit dem `amqspubc` -Beispiel und der `amqssubc` -Stichprobe getestet werden. Weitere Informationen zu diesen Beispielen finden Sie im Abschnitt Publish/Subscribe-Beispielprogramme .

1. Öffnen Sie zwei Befehlszeilenfenster. Die erste Befehlszeile ist für das `amqspubc` -Publizierungsbeispiel und die zweite Befehlszeile für die `amqssubc` -Abonnementsprobe.
2. Geben Sie den folgenden Befehl in die Befehlszeile ein:

   ```
   amqspubc Sports QM1
   ```

   Hierbei steht *Sports* für die Themenzeichenfolge des in einem früheren Beispiel definierten TOPIC-Objekts und *QM1* für den Namen des Warteschlangenmanagers.
3. Geben Sie den folgenden Befehl in die Befehlszeile ein: 2:

   ```
   amqssubc Sports QM1
   ```

   Dabei sind *Sports* und *QM1* die gleichen wie in Schritt „2” auf Seite 159.
4. Geben Sie `Hello world` in Befehlszeile 1 ein. Wenn der Port und die IP-Adresse, die im COMMINFO-Objekt angegeben sind, ordnungsgemäß konfiguriert sind, gibt das `amqssubc` -Beispiel, das am Port für Veröffentlichungen von der angegebenen Adresse empfangsbereit ist, `Hello world` in der Befehlszeile 2 aus.

IBM WebSphere MQ Multicast-Thementopologie

Dieses Beispiel soll das Verständnis der IBM WebSphere MQ Multicast-Thementopologie erleichtern.

Die IBM WebSphere MQ Multicast-Unterstützung erfordert, dass jede Unterverzeichnisstruktur eine eigene Multicastgruppe und einen eigenen Datenstrom in der gesamten Hierarchie hat.

Das IP-Adressierungsschema *classful network* umfasst einen designierten Adressraum für die Multicastadresse. Der vollständige IP-Adressenbereich für Multicasting liegt zwischen `224.0.0.0` und `239.255.255.255`, allerdings sind einige dieser Adressen reserviert. Eine Liste der reservierten Adressen erhalten Sie von Ihrem Systemadministrator; Sie finden sie außerdem unter IPv4 Multicast Address Space Registry. Es wird empfohlen, den lokalen Multicastadressbereich von `239.0.0.0` bis `239.255.255.255` zu verwenden.

Im folgenden Diagramm gibt es zwei mögliche Multicastdatenströme:

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

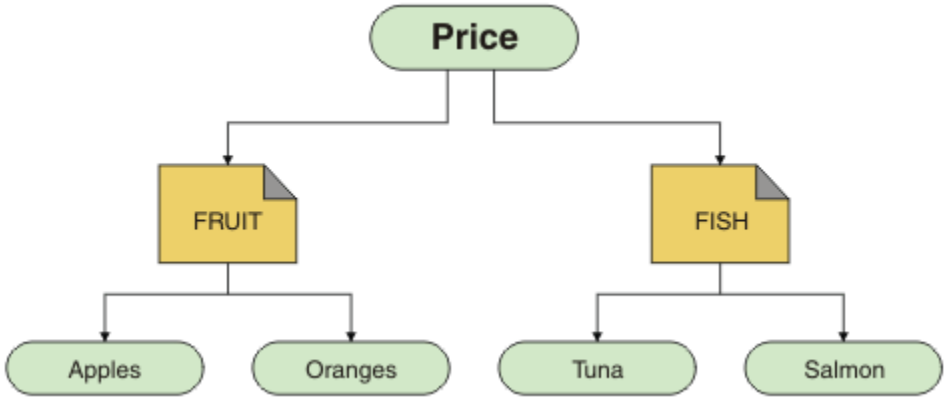
Dabei sind *239.XXX.XXX.XXX* und *239.YYY.YYY.YYY* gültige Multicastadressen.

Diese Themendefinitionen werden verwendet, um eine Themenstruktur zu erstellen, wie im folgenden Diagramm dargestellt:

```

DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)

DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Jedes Multicast-Kommunikationsinformationsobjekt (COMMINFO) stellt einen anderen Datenstrom dar, da ihre Gruppenadressen unterschiedlich sind. In diesem Beispiel ist das Thema FRUIT für die Verwendung des COMMINFO-Objekts MC1definiert, das Thema FISH ist für die Verwendung des COMMINFO-Objekts MC2definiert und der Price -Knoten hat keine Multicastdefinitionen.

In WebSphere MQ Multicast ist die Länge der Themenzeichenfolgen auf 255 Zeichen begrenzt. Diese Einschränkung bedeutet, dass bei den Namen von Knoten und Blattknoten in der Baumstruktur Vorsicht geboten ist. Wenn die Namen von Knoten und Blattknoten zu lang sind, ist die Themenzeichenfolge möglicherweise länger als 255 Zeichen und gibt den Ursachencode 2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR zurück. Es wird empfohlen, Themenzeichenfolgen so kurz wie möglich zu machen, da längere Themenzeichenfolgen sich nachteilig auf die Leistung auswirken können.

Größe von Multicastnachrichten steuern

In diesem Abschnitt erfahren Sie mehr über das Nachrichtenformat WebSphere MQ und können die Größe von WebSphere MQ -Nachrichten reduzieren.

WebSphere MQ -Nachrichten ist eine Reihe von Attributen zugeordnet, die im Nachrichtendeskriptor enthalten sind. Bei kleinen Nachrichten stellen diese Attribute den größten Teil des Datenverkehrs dar und können erhebliche negative Auswirkungen auf die Übertragungsgeschwindigkeit haben. WebSphere MQ Multicast ermöglicht es dem Benutzer, zu konfigurieren, welche dieser Attribute gegebenenfalls zusammen mit der Nachricht übertragen werden.

Das Vorhandensein von Nachrichtenattributen, die keine Themenzeichenfolge sind, hängt davon ab, ob das Objekt COMMINFO angibt, dass sie gesendet werden müssen oder nicht. Wenn ein Attribut nicht übertragen wird, wendet die empfangende Anwendung einen Standardwert an. Die MQMD-Standardwerte entsprechen nicht unbedingt dem MQMD_DEFAULT-Wert und werden in Tabelle 9 auf Seite 161beschrieben.

Das COMMINFO-Objekt enthält das Attribut MCPROP , das steuert, wie viele MQMD-Felder und Benutzereigenschaften mit der Nachricht fließen. Wenn Sie den Wert dieses Attributs auf eine geeignete Stufe setzen, können Sie die Größe der WebSphere MQ Multicast-Nachrichten steuern:

MCPROP
Die Multicasteigenschaften steuern, wie viele der MQMD-Eigenschaften und Benutzereigenschaften mit der Nachricht fließen.

ALL
Alle Benutzereigenschaften und alle Felder des MQMD werden übertragen.

REPLY
Nur Benutzereigenschaften und MQMD-Felder, die sich auf die Beantwortung der Nachrichten beziehen, werden übertragen. Diese Eigenschaften sind:

- MsgType
- MessageId
- CorrelId

- ReplyToQ
- ReplyToQmgr

BENUTZER
Es werden nur die Benutzereigenschaften übertragen.

KEINE
Es werden keine Benutzereigenschaften oder MQMD-Felder übertragen.

COMPAT
Dieser Wert bewirkt, dass die Übertragung der Nachricht in einem kompatiblen Modus an RMMer-folgt, was eine gewisse Interoperation mit den aktuellen XMS -Anwendungen und WebSphere Message Broker RMM -Anwendungen ermöglicht.

Multicastnachrichten-Attribute

Nachrichtenattribute können aus verschiedenen Bereichen stammen, wie z. B. MQMD, die Felder in den MQRFH2-und Nachrichteneigenschaften.

Die folgende Tabelle zeigt, was geschieht, wenn Nachrichten gemäß dem Wert von MCPROP gesendet werden und der Standardwert verwendet wird, wenn ein Attribut nicht gesendet wird.

Tabelle 9. Messaging-Attribute und ihre Beziehungen zu Multicasting

Attribut	**Aktion bei Verwendung von Multicasting**	**Standardwert, wenn nicht übertragen**
TopicString	Immer eingeschlossen	Nicht zutreffend
MQMQ-Struktur-ID	Nicht übertragen	Nicht zutreffend
MQMD-Version	Nicht übertragen	Nicht zutreffend
Bericht	Wird eingeschlossen, wenn nicht der Standardwert	0
MsgType	Wird eingeschlossen, wenn nicht der Standardwert	MQMT_DATAGRAM
Verfall	Wird eingeschlossen, wenn nicht der Standardwert	0
Feedback	Wird eingeschlossen, wenn nicht der Standardwert	0
Encoding	Wird eingeschlossen, wenn nicht der Standardwert	MQENC_NORMAL (equiv)
CodedCharSetId	Wird eingeschlossen, wenn nicht der Standardwert	1208
Format	Wird eingeschlossen, wenn nicht der Standardwert	MQRFH2
Priority	Wird eingeschlossen, wenn nicht der Standardwert	4
Permanenz	Wird eingeschlossen, wenn nicht der Standardwert	MQPER_NOT_PERSISTENT
MsgId	Wird eingeschlossen, wenn nicht der Standardwert	Null
CorrelId	Wird eingeschlossen, wenn nicht der Standardwert	Null

Tabelle 9. Messaging-Attribute und ihre Beziehungen zu Multicasting (Forts.)		
Attribut	**Aktion bei Verwendung von Multicasting**	**Standardwert, wenn nicht übertragen**
BackoutCount	Wird eingeschlossen, wenn nicht der Standardwert	0
ReplyToQ	Wird eingeschlossen, wenn nicht der Standardwert	Leer
ReplyToQMgr	Wird eingeschlossen, wenn nicht der Standardwert	Leer
UserIdentifier	Wird eingeschlossen, wenn nicht der Standardwert	Leer
AccountingToken	Wird eingeschlossen, wenn nicht der Standardwert	Null
PutAppIType	Wird eingeschlossen, wenn nicht der Standardwert	MQAT_JAVA
PutAppIName	Wird eingeschlossen, wenn nicht der Standardwert	Leer
PutDate	Wird eingeschlossen, wenn nicht der Standardwert	Leer
PutTime	Wird eingeschlossen, wenn nicht der Standardwert	Leer
ApplOriginData	Wird eingeschlossen, wenn nicht der Standardwert	Leer
GroupID	Ausgeschlossen	Nicht zutreffend
MsgSeqNumber	Ausgeschlossen	Nicht zutreffend
Offset	Ausgeschlossen	Nicht zutreffend
MsgFlags	Ausgeschlossen	Nicht zutreffend
OriginalLength	Ausgeschlossen	Nicht zutreffend
UserProperties	Eingeschlossen	Nicht zutreffend

Zugehörige Verweise
ALTER COMMINFO
DEFINE COMMINFO

Datenkonvertierung für Multicast-Messaging aktivieren

In diesen Informationen erfahren Sie, wie die Datenkonvertierung für die WebSphere MQ Multicast-Nachrichtenübertragung funktioniert.

Bei WebSphere MQ Multicast handelt es sich um ein gemeinsam genutztes verbindungsunabhängiges Protokoll und daher kann nicht jeder Client spezielle Anforderungen zur Datenkonvertierung stellen. Jeder Client, der den gleichen Multicastdatenstrom subskribiert, empfängt die gleichen Binärdaten; wenn eine Konvertierung von WebSphere MQ-Daten erforderlich ist, wird die Konvertierung deshalb lokal auf jedem Client ausgeführt.

In einer heterogenen Plattforminstallation kann es sein, dass die meisten Clients die Daten in einem Format benötigen, das nicht das native Format der Übertragungsanwendung ist. In dieser Situation

können die **CCSID** -und **ENCODING** -Werte des Multicast- COMMINFO -Objekts verwendet werden, um die Codierung der Nachrichtenübertragung für Effizienz zu definieren.

WebSphere MQ Multicast unterstützt die Datenkonvertierung der Nachrichtennutzdaten für die folgenden integrierten Formate:

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

Zusätzlich zu diesen Formaten können Sie auch eigene Formate definieren und einen Datenkonvertierungsexit MQDXP-Data-Conversion Exit Parameter verwenden.

Informationen zur Programmierung von Datenkonvertierungen finden Sie unter Datenkonvertierung in der MQI für Multicast-Messaging .

Weitere Informationen zur Datenkonvertierung finden Sie unter Datenkonvertierung .

Weitere Informationen zu Datenkonvertierungsexits und `ClientExitPath`finden Sie im Abschnitt ZeilengruppeClientExitder Clientkonfigurationsdatei.

Multicast-Anwendungsüberwachung

In diesem Abschnitt finden Sie Informationen zur Verwaltung und Überwachung von WebSphere MQ Multicast.

Der Status der aktuellen Publisher und Subskribenten für den Multicast-Datenverkehr (z. B. die Anzahl der gesendeten und empfangenen Nachrichten oder die Anzahl der verlorenen Nachrichten) wird regelmäßig vom Client an den Server übertragen. Wenn der Status empfangen wird, gibt das Attribut COMMEV des COMMINFO-Objekts an, ob der Warteschlangenmanager eine Ereignisnachricht in die Warteschlange SYSTEM.ADMIN.PUBSUB.EVENT einreiht. Die Ereignisnachricht enthält die empfangenen Statusinformationen. Diese Informationen sind eine unschätzbare Diagnose-Hilfe bei der Suche nach der Ursache eines Problems.

Mit dem MQSC-Befehl **DISPLAY CONN** können Sie Verbindungsinformationen zu den Anwendungen anzeigen, die mit dem Warteschlangenmanager verbunden sind. Weitere Informationen zum Befehl **DISPLAY CONN** finden Sie unter DISPLAY CONN.

Verwenden Sie den MQSC-Befehl **DISPLAY TPSTATUS** , um den -Status Ihrer Publisher und Subskribenten anzuzeigen. Weitere Informationen zum Befehl **DISPLAY TPSTATUS** finden Sie unter DISPLAY TPSTATUS.

COMMEV und der Anzeiger für die Zuverlässigkeit der Multicastnachricht

Der *Zuverlässigkeitsanzeiger*, der zusammen mit dem Attribut **COMMEV** des COMMINFO-Objekts verwendet wird, ist ein Schlüsselelement bei der Überwachung von WebSphere MQ Multicast-Publishern und -Subskribenten. Der Zuverlässigkeitsindikator (das Feld **MSGREL** , das in den Publish/Subscribe-Statusbefehlen zurückgegeben wird) ist ein WebSphere MQ -Indikator, der den Prozentsatz der Übertragungen darstellt, bei denen keine Fehler aufgetreten sind. Manchmal müssen Nachrichten aufgrund eines Übertragungsfehlers erneut übertragen werden. Dies spiegelt sich im Wert von **MSGREL**wider. Mögliche Ursachen von Übertragungsfehlern sind langsame Subskribenten, ausgelastende Netze und Netzausfälle. **COMMEV** steuert, ob Ereignisnachrichten für Multicastkennungen generiert werden, die mit dem Objekt COMMINFO erstellt werden, und wird auf einen von drei möglichen Werten gesetzt:

INAKTIVIERT
Ereignisnachrichten werden nicht geschrieben.

ENABLED
Ereignisnachrichten werden immer mit einer Häufigkeit geschrieben, die im Parameter COMMINFO **MONINT** definiert ist.

EXCEPTION
Ereignisnachrichten werden erstellt, wenn die Nachrichtenzuverlässigkeit unter dem Grenzwert für die Zuverlässigkeit liegt. Eine Nachrichtenzuverlässigkeitsstufe von 90% oder weniger zeigt an, dass ein Problem mit der Netzkonfiguration aufgetreten ist oder dass eine oder mehrere der Publish/Subscribe-Anwendungen zu langsam ausgeführt werden:

- Der Wert **MSGREL(100,100)** gibt an, dass es weder in der Kurzzeit noch im Langzeitzeitrahmen zu Problemen gekommen ist.
- Der Wert **MSGREL(80,60)** gibt an, dass 20% der Nachrichten derzeit Probleme haben, aber es ist auch eine Verbesserung für den Langzeitwert von 60.

Clients können die Übertragung und den Empfang von Multicastverkehr auch dann fortsetzen, wenn die Unicast-Verbindung zum Warteschlangenmanager unterbrochen ist. Daher sind die Daten möglicherweise nicht auf dem neuesten Stand.

Multicast-Nachrichtenzuverlässigkeit

In diesem Abschnitt erfahren Sie, wie Sie die WebSphere MQ Multicast-Subskription und das Nachrichtenprotokoll festlegen.

Ein Schlüsselelement zur Überwindung von Übertragungsfehlern mit Multicasting ist die Pufferung von übertragenen Daten durch WebSphere MQ(ein Protokoll der Nachrichten, die am Übertragungsende der Verbindung aufbewahrt werden sollen). Dieser Prozess bedeutet, dass keine Pufferung von Nachrichten im einreihenden Anwendungsprozess erforderlich ist, da WebSphere MQ die Zuverlässigkeit bietet. Die Größe dieses Protokolls wird über das Kommunikationsinformationsobjekt (COMMINFO) konfiguriert, wie in den folgenden Informationen beschrieben. Ein größerer Übertragungspuffer bedeutet, dass bei Bedarf mehr Übertragungsprotokoll übertragen wird, aber aufgrund der Art der Multicast-Übertragung kann eine 100% garantierten Zustellung nicht unterstützt werden.

Das WebSphere MQ Multicast-Nachrichtenprotokoll wird im Kommunikationsinformationsobjekt (COMMINFO) durch das Attribut **MSGHIST** gesteuert:

MSGHIST
Dieser Wert ist die Menge an Nachrichtenprotokollen in Kilobyte, die vom System zur Bearbeitung erneuter Übertragungen im Falle negativer Rückmeldungen behalten wird.

Der Wert `0` gibt den niedrigsten Grad an Zuverlässigkeit an. Der Standardwert ist `100` KB.

Das neue WebSphere MQ Multicast-Subskriptionsprotokoll wird im Kommunikationsinformationsobjekt (COMMINFO) durch das Attribut **NSUBHIST** gesteuert:

NSUBHIST
Das neue Abonnentenprotokoll steuert, ob ein Abonnent, der an einem Veröffentlichungs-Stream teilnimmt, so viele Daten wie aktuell verfügbar empfängt, oder ob er nur Veröffentlichungen empfängt, die seit dem Zeitpunkt der Subskription erstellt wurden.

KEINE
Der Wert `NONE` bewirkt, dass der Sender nur Veröffentlichungen überträgt, die seit dem Zeitpunkt der Subskription erstellt wurden. `NONE` ist der Standardwert.

ALLE
Der Wert `ALL` bewirkt, dass der Sender den gesamten bekannten Verlauf eines Themas erneut überträgt. Unter bestimmten Umständen kann diese Situation ein ähnliches Verhalten für ständige Veröffentlichungen liefern.

Anmerkung: Die Verwendung des Werts `ALL` kann sich unter Umständen bei einem umfangreichen Themenverlauf nachteilig auf die Leistung auswirken, da der gesamte Verlauf erneut übertragen wird.

Zugehörige Verweise
DEFINE COMMINFO
ALTER COMMINFO

Erweiterte Multicast-Tasks

In diesem Abschnitt erfahren Sie mehr über erweiterte WebSphere MQ Multicast-Verwaltungstasks wie die Konfiguration von `.ini` -Dateien und die Interoperabilität mit WebSphere MQ LLM.

Hinweise zur Sicherheit in einer Multicast-Installation finden Sie unter Multicast-Sicherheit .

Überbrückung zwischen Multicast-und Nicht-Multicast-Publish/Subscribe-Domänen

Verwenden Sie diese Informationen, um zu verstehen, was geschieht, wenn ein Nicht-Multicast-Publisher in einem WebSphere MQ Multicast-fähigen Thema veröffentlicht.

Wenn ein Nicht-Multicast-Publisher in einem Thema veröffentlicht, das als **MCAST** aktiviert ist und **BRIDGE** aktiviert ist, überträgt der Warteschlangenmanager die Nachricht direkt an alle Subskribenten, die empfangsbereit sein könnten, über Multicasting. Ein Multicast-Publisher kann nicht in Themen veröffentlichen, die nicht Multicasting-fähig sind.

Vorhandene Themen können Multicasting-fähig sein, indem Sie die Parameter **MCAST** und **COMMINFO** eines Themenobjekts festlegen. Weitere Informationen zu diesen Parametern finden Sie unter Ursprüngliche Multicastkonzepte .

Das Attribut COMMINFO-Objekt **BRIDGE** steuert Veröffentlichungen von Anwendungen, die keine Multicast-Anwendungen verwenden. Wenn **BRIDGE** auf ENABLED gesetzt ist und der Parameter **MCAST** des Themas ebenfalls auf ENABLED gesetzt ist, werden Veröffentlichungen von Anwendungen, die nicht Multicast verwenden, zu Anwendungen überbrückt, die nicht verwendet werden. Weitere Informationen zum Parameter **BRIDGE** finden Sie in DEFINE COMMINFO .

INI-Dateien für Multicasting konfigurieren

Verwenden Sie diese Informationen, um die WebSphere MQ Multicast-Felder in den `.ini` -Dateien zu verstehen.

Zusätzliche WebSphere MQ Multicast-Konfiguration kann in einer `ini` -Datei vorgenommen werden. Die spezielle `ini` -Datei, die Sie verwenden müssen, hängt von der Art der Anwendung ab:

- Client: Konfigurieren Sie die *MQ_DATA_PATH*`/mqclient.ini` -Datei.
- Warteschlangenmanager: Konfigurieren Sie die Datei *MQ_DATA_PATH*`/qmgrs/`*QMNAME*`/qm.ini` .

Dabei steht *MQ_DATA_PATH* für die Speicherposition des Datenverzeichnisses von WebSphere MQ (`/var/mqm/mqclient.ini`) und *QMNAME* für den Namen des Warteschlangenmanagers, für den die Datei `.ini` gilt.

Die Datei `.ini` enthält Felder, mit denen das Verhalten von WebSphere MQ Multicast optimiert werden kann:

```
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
```

Protokoll

UDP
In diesem Modus werden Pakete mit Hilfe des UDP-Protokolls gesendet. Netzelemente können keine Unterstützung in der Multicastverteilung bereitstellen, wie sie im IP-Modus ausgeführt werden. Das Paketformat bleibt mit PGM kompatibel. Dies ist der Standardwert.

IP
In diesem Modus sendet der Sender unformatierte IP-Pakete. Netzelemente mit PGM-Unterstützung helfen bei der zuverlässigen Multicastpaketverteilung. Dieser Modus ist vollständig kompatibel mit dem PGM-Standard.

IPVersion

IPV4
Kommunikation nur über das Protokoll IPv4 . Dies ist der Standardwert.

IPV6
Kommunikation nur über das IPv6 -Protokoll.

ANY
Kommunizieren Sie mit IPv4und/oder IPv6, je nachdem, welches Protokoll verfügbar ist.

BEIDE
Unterstützt die Kommunikation sowohl über IPv4 als auch über IPv6.

LimitTransRate

INAKTIVIERT
Es gibt keine Übertragungsraten-Steuerung. Dies ist der Standardwert.

STATISCH
Implementiert die Steuerung der statischen Übertragungsgeschwindigkeit. Der Sender würde nicht mit einer Rate übertragen, die die durch den Parameter TransRateLimit angegebene Rate überschreitet.

DYNAMIC
Der Sender passt seine Übertragungsrate entsprechend dem Feedback, das es von den Empfängern erhält, an. In diesem Fall darf die Übertragungsgeschwindigkeit nicht größer sein als der durch den Parameter TransRateLimit angegebene Wert. Der Sender versucht, eine optimale Übertragungsrate zu erreichen.

TransRateLimit
Die Übertragungsgeschwindigkeit in Kbps.

SocketTTL
Der Wert von SocketTTL bestimmt, ob der Multicast-Datenverkehr einen Router passieren kann, oder die Anzahl der Router, die er passieren kann.

Batch
Steuert, ob Nachrichten gesendet oder sofort gesendet werden. Es gibt zwei mögliche Werte:

- *NO* Die Nachrichten werden nicht in der Stapelverarbeitung gesendet, sondern sofort gesendet.
- *YES* Die Nachrichten werden stapelweise ausgeführt.

Schleife
Setzen Sie den Wert auf 1 , um die Multicastschleife zu aktivieren. Die Multicastschleife definiert, ob die gesendeten Daten an den Host zurückgeschleift werden oder nicht.

Schnittstelle
Die IP-Adresse der Schnittstelle, auf der Multicastdatenverkehr fließt. Weitere Informationen und Hinweise zur Fehlerbehebung finden Sie in den Abschnitten Multicastanwendungen auf einem Nicht-Multicast-Netz testen und Geeignetes Netz für Multicastverkehr festlegen.

FeedbackMode

NACK
Rückmeldung durch negative Bestätigungen. Dies ist der Standardwert.

ACK
Rückmeldung durch positive Bestätigungen.

WAIT1
Rückmeldungen durch positive Bestätigungen, bei denen der Sender nur auf 1 ACK von einem der Empfänger wartet.

HeartbeatTimeout
Das Zeitlimit für die Überwachungssignale in Millisekunden. Der Wert 0 gibt an, dass die Ereignisse des Heartbeat-Zeitlimits vom Empfänger oder Empfänger des Themas nicht ausgelöst werden. Der Standardwert ist 20000.

HeartbeatInterval
Das Intervall der Überwachungssignale in Millisekunden. Der Wert 0 gibt an, dass keine Überwachungssignale gesendet werden. Das Intervall der Überwachungssignale muss deutlich kleiner als der Wert von **`HeartbeatTimeout`** sein, um falsche Zeitlimitüberschreitungen bei Überwachungssignalen zu vermeiden. Der Standardwert ist 2000.

Multicast-Interoperabilität mit WebSphere MQ Low Latency Messaging

Verwenden Sie diese Informationen, um die Interoperabilität zwischen WebSphere MQ Multicast und WebSphere MQ Low Latency Messaging (LLM) zu verstehen.

Eine einfache Übertragung von Nutzdaten ist für eine Anwendung mit LLM möglich, wobei eine andere Anwendung Multicast verwendet, um Nachrichten in beide Richtungen auszutauschen. Obwohl Multicast die LLM-Technologie verwendet, ist das LLM-Produkt selbst nicht eingebettet. Daher ist es möglich, sowohl LLM als auch WebSphere MQ Multicast zu installieren und die beiden Produkte separat zu betreiben und zu bedienen.

LLM-Anwendungen, die mit Multicasting kommunizieren, müssen möglicherweise Nachrichteneigenschaften senden und empfangen. Die WebSphere MQ -Nachrichteneigenschaften und MQMD-Felder werden als LLM-Nachrichteneigenschaften mit bestimmten LLM-Nachrichteneigenschaftscodes übertragen, wie in der folgenden Tabelle dargestellt:

Tabelle 10. Zuordnungen von WebSphere MQ -Nachrichteneigenschaften zu WebSphere MQ -LLM-Eigenschaften

WebSphere MQ-Eigenschaft	WebSphere MQ LLM-Eigenschaftstyp	LLM-Eigenschaftenart	LLM-Eigenschaftscode
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_String	-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_String	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1020

Tabelle 10. Zuordnungen von WebSphere MQ -Nachrichteneigenschaften zu WebSphere MQ -LLM-Eigenschaften (Forts.)			
WebSphere MQ-Eigenschaft	**WebSphere MQ LLM-Eigenschaftstyp**	**LLM-Eigenschaftenart**	**LLM-Eigenschaftscode**
MQMD.PutTime	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

Weitere Informationen zu LLM finden Sie in der LLM-Produktdokumentation: WebSphere MQ Low Latency Messaging.

VerwaltungHP Integrity NonStop Server

In diesem Abschnitt finden Sie Informationen zu Verwaltungstasks für den IBM WebSphere MQ -Client für HP Integrity NonStop Server.

Es stehen Ihnen zwei Verwaltungstasks zur Verfügung:

1. Manuelles Starten des TMF/Gateways von Pathway aus.
2. Das TMF/Gateway von der Pfadbahn aus stoppen.

Manuelles Starten des TMF/Gateways von Pathway

Sie können Pathway erlauben, das TMF/Gateway automatisch in der ersten Aufforderung zur Registrierung zu starten, oder Sie können das TMF/Gateway manuell von der Pathway-Datei aus starten.

Vorgehensweise

Geben Sie den folgenden PATHCOM-Befehl ein, um das TMF/Gateway manuell von der Pathway-Datei zu starten:

```
START SERVER <server_class_name>
```

Wenn eine Clientanwendung eine Anfrage für die Registrierung vor dem TMF/Gateway beendet, die Wiederherstellung von unbestätigungsfreien Transaktionen abgeschlossen hat, wird die Anforderung für bis zu 1 Sekunde gehalten. Wenn die Wiederherstellung innerhalb dieser Zeit nicht abgeschlossen wird, wird die Registrierung zurückgewiesen. Der Client empfängt dann einen MQRC_UOW_ENLISTMENT_ERROR-Fehler von der Verwendung einer transaktionsorientierten MQI.

Stoppen des TMF/Gateways von Pathway

In dieser Übung wird beschrieben, wie der TMF/Gateway von Pathway aus gestoppt wird und wie der TMF/Gateway nach dem Stoppen erneut gestartet wird.

Vorgehensweise

1. Geben Sie den folgenden Befehl ein, um zu verhindern, dass neue Listenanforderungen an das TMF/Gateway gestellt werden:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. Geben Sie den folgenden Befehl ein, um den TMF/Gateway auszulösen, um alle In-Flight-Operationen auszuführen und zu beenden:

   ```
   STOP SERVER <server_class_name>
   ```

3. Geben Sie den folgenden Befehl ein, um zu ermöglichen, dass das TMF/Gateway entweder automatisch bei der ersten Registrierung oder nach den Schritten 1 und 2 erneut gestartet wird:

   ```
   THAW SERVER <server_class_name>
   ```

   Anwendungen werden daran gehindert, neue Einreihungsanforderungen zu stellen, und es ist nicht möglich, den Befehl **START** abzusetzen, bis Sie den Befehl **THAW** absetzen.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter als IBM liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

Bei Lizenzanforderungen zu Double-Byte-Information (DBCS) wenden Sie sich bitte an die IBM Abteilung für geistiges Eigentum in Ihrem Land oder senden Sie Anfragen schriftlich an folgende Adresse:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan, Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in dieser Veröffentlichung werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter alsIBM werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation Software Interoperability Coordinator, Abteilung 49XA 3605 Highway 52 N Rochester, MN 55901 U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedin-

gungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Die in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen vonIBM den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann die Genauigkeit von Leistung, Kompatibilität oder anderen Merkmalen, die sich auf Nicht-IBM-Produkte beziehen, nicht bestätigen. Fragen zu den Leistungsmerkmalen von Nicht-IBM-Produkten sind an die Anbieter dieser Produkte zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Um diese so realistisch wie möglich zu gestalten, enthalten sie auch Namen von Personen, Firmen, Marken und Produkten. Sämtliche dieser Namen sind fiktiv. Ähnlichkeiten mit Namen und Adressen tatsächlicher Unternehmen oder Personen sind zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musterprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Musterprogramme kostenlos (d. h. ohne Zahlung an IBM) kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Informationen zu Programmierschnittstellen

Die bereitgestellten Informationen zur Programmierschnittstelle sollen Sie bei der Erstellung von Anwendungssoftware für dieses Programm unterstützen.

Dieses Handbuch enthält Informationen zu geplanten Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zum Abrufen der Services von IBM WebSphere MQzu schreiben.

Diese Informationen können jedoch auch Angaben über Diagnose, Bearbeitung und Optimierung enthalten. Die Informationen zu Diagnose, Bearbeitung und Optimierung sollten Ihnen bei der Fehlerbehebung für die Anwendungssoftware helfen.

Wichtig: Verwenden Sie diese Diagnose-, Änderungs-und Optimierungsinformationen nicht als Programmierschnittstelle, da sie Änderungen unterliegen.

Marken

IBM, das IBM Logo, ibm.com, sind Marken der IBM Corporation in den USA und/oder anderen Ländern. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Microsoft und Windows sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke der The Open Group in den USA und/oder anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Dieses Produkt enthält Software, die von Eclipse Project (http://www.eclipse.org/) entwickelt wurde.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Teilenummer:

(1P) P/N: