7.5

Administrowanie produktem IBM WebSphere MQ

Uwaga

Przed skorzystaniem z niniejszych informacji oraz produktu, którego one dotyczą, należy zapoznać się z informacjami zamieszczonymi w sekcji “Uwagi” na stronie 167.

Niniejsze wydanie dotyczy wersji 7 wydanie 5 produktu IBM® WebSphere MQ oraz wszystkich kolejnych wydań i modyfikacji, o ile nie zostanie to określone inaczej w nowych wydaniach.

Wysyłając informacje do IBM, użytkownik przyznaje IBM niewyłączne prawo do używania i rozpowszechniania informacji w dowolny sposób, jaki uzna za właściwy, bez żadnych zobowiązań wobec ich autora.

© Copyright International Business Machines Corporation 2007, 2024.

Spis treści

Administrowanie..... 5
Administrowanie lokalne i zdalne.....7
Sposób korzystania z komend sterujących produktu IBM WebSphere MQ..... 8
Automatyzacja zadań administracyjnych.....8
Wprowadzenie do formatów komend programowalnych.....9
Korzystanie z interfejsu MQAI w celu uproszczenia korzystania z systemów PCF.....20
Wprowadzenie do interfejsu IBM WebSphere MQ Administration Interface (MQAI).....20
Interfejs administracyjny produktu IBM WebSphere MQ (MQAI).....22
Administrowanie przy użyciu programu IBM WebSphere MQ Explorer.....58
Co można zrobić za pomocą programu IBM WebSphere MQ Explorer.....59
Konfigurowanie programu IBM WebSphere MQ Explorer.....61
Zabezpieczenia w systemie Windows.....67
Rozszerzanie programu IBM WebSphere MQ Explorer (tylko platformy Windows i Linux na platformie x86).....70
Korzystanie z aplikacji paska zadań IBM WebSphere MQ (tylko w systemie Windows)..... 70
Aplikacja monitora alertów produktu IBM WebSphere MQ (tylko system Windows)..... 71
Administrowanie lokalnymi obiektami IBM WebSphere MQ.....71
Uruchamianie i zatrzymywanie menedżera kolejek..... 71
Ręczne zatrzymywanie menedżerów kolejek.....73
Wykonywanie lokalnych zadań administracyjnych za pomocą komend MQSC.....75
Praca z menedżerami kolejek..... 84
Praca z kolejkami lokalnymi.....86
Praca z kolejkami aliasami..... 91
Praca z kolejkami modelami..... 93
Praca z tematami administracyjnymi.....93
Praca z subskrypcjami.....96
Praca z usługami.....99
Zarządzanie obiektami w celu wyzwolenia..... 106
Administrowanie zdalnymi obiektami produktu IBM WebSphere MQ.....108
Kanały, klastry i kolejkowanie zdalne..... 108
Zdalne administrowanie z lokalnego menedżera kolejek.....110
Tworzenie lokalnej definicji kolejki zdalnej..... 116
Używanie definicji kolejek zdalnych jako aliasów..... 118
Konwersja danych..... 119
Administrowanie produktem IBM WebSphere MQ Telemetry..... 121
Konfigurowanie menedżera kolejek dla telemetrii w systemach Linux i AIX.....122
Konfigurowanie menedżera kolejek dla telemetrii w systemie Windows..... 123
Konfigurowanie menedżera kolejek w celu wysyłania komunikatów do klientów MQTT..... 125
Identyfikacja klienta, autoryzacja i uwierzytelnianie.....128
Uwierzytelnianie kanału telemetrycznego przy użyciu protokołu SSL..... 135
Prywatność publikacji przy użyciu SSL.....137
konfiguracja SSL.....138
Konfiguracja JAAS.....143
Pojęcia dotyczące urządzeń demona IBM WebSphere MQ Telemetry.....145
Administrowanie rozgłaszanie..... 156
Pierwsze kroki w trybie rozsyłania grupowego..... 156
IBM WebSphere MQ Multicast, topologia tematu.....157
Zmniejszanie wielkości komunikatów rozsyłania grupowego..... 158
Włączanie konwersji danych na potrzeby przesyłania komunikatów w trybie Multicast.....160
Administrowanie i monitorowanie rozsyłania grupowego..... 161
Ustawianie historii komunikatów subskrypcji rozsyłania grupowego..... 162
Zaawansowane zadania rozsyłania grupowego..... 162

Administrowanie serwerem HP Integrity NonStop Server....166
Samodzielne uruchamianie programu TMF/Gateway z ścieżki....166
Zatrzymywanie programu TMF/Gateway z Pathway....166

Uwagi....167
Informacje dotyczące interfejsu programistycznego....168
Znaki towarowe....168

administrowanieIBM WebSphere MQ

Administrowanie menedżerami kolejek i powiązanymi zasobami obejmuje zadania, które często są wykonywane w celu aktywowania zasobów i zarządzania nimi. Wybierz metodę administrowania menedżerami kolejek i powiązanymi zasobami.

Obiekty produktu IBM WebSphere MQ można administrować lokalnie lub zdalnie, patrz “Administrowanie lokalne i zdalne” na stronie 7.

Istnieje wiele różnych metod tworzenia i administrowania menedżerami kolejek i ich zasobami pokrewnymi w produkcie IBM WebSphere MQ. Do metod tych należą interfejsy wiersza komend, graficzny interfejs użytkownika oraz interfejs API administrowania. Więcej informacji na temat każdego z tych interfejsów można znaleźć w sekcjach i odsyłaczami w tym temacie.

Istnieją różne zestawy komend, których można używać do administrowania produktem IBM WebSphere MQ w zależności od platformy:

- “Komendy sterujące produktu IBM WebSphere MQ” na stronie 5
- “Komendy produktu IBM WebSphere MQ Script (MQSC)” na stronie 5
- “Programowalne formaty komend (PCFs)” na stronie 6

Istnieją również inne następujące opcje tworzenia obiektów IBM WebSphere MQ i zarządzania nimi:

- “IBM WebSphere MQ Explorer” na stronie 6
- “Aplikacja Konfiguracja domyślna produktu Windows” na stronie 7
- “Usługa Microsoft Cluster Service (MSCS)” na stronie 7

Niektóre zadania administrowania i monitorowania można zautomatyzować zarówno dla lokalnych, jak i zdalnych menedżerów kolejek za pomocą komend PCF. Komendy te można również uprościć przy użyciu interfejsu administracyjnego produktu IBM WebSphere MQ (MQAI) na niektórych platformach. Więcej informacji na temat automatyzowania zadań administracyjnych zawiera sekcja “Automatyzacja zadań administracyjnych” na stronie 8.

Komendy sterujące produktu IBM WebSphere MQ

Komendy sterujące umożliwiają samodzielne wykonywanie zadań administracyjnych w menedżerach kolejek.

IBM WebSphere MQ dla systemów Windows, UNIX and Linux® udostępnia *komendy sterujące* , które są dostępne w wierszu komend systemu.

Komendy sterujące są opisane w sekcji Tworzenie i zarządzanie menedżerami kolejek. Informacje na temat komend sterujących można znaleźć w sekcji Komendy sterujące programuIBM WebSphere MQ.

Komendy produktu IBM WebSphere MQ Script (MQSC)

Za pomocą komend MQSC służą do zarządzania obiektami menedżera kolejek, w tym samym menedżerem kolejek, kolejkami, definicjami procesów, listami nazw, kanałami, kanałami połączeń klientów, obiektami nasłuchiwania, usługami i obiektami informacji uwierzytelniających.

Komendy MQSC są emitowane do menedżera kolejek za pomocą komendy `runmqsc` . Można to zrobić interaktywnie, wydając komendy z klawiatury lub można przekierować standardowe urządzenie wejściowe (stdin), aby uruchomić sekwencję komend z pliku tekstowego ASCII. W obu przypadkach format komend jest taki sam.

Komendę `runmqsc` można uruchomić w trzech trybach, w zależności od opcji ustawionych w komendzie:

- *Tryb weryfikacji*, w którym komendy MQSC są weryfikowane w lokalnym menedżerze kolejek, ale nie są uruchamiane.

© Copyright IBM Corp. 2007, 2024

- *Tryb bezpośredni*, w którym komendy MQSC są uruchamiane w lokalnym menedżerze kolejek
- *Tryb pośredni*, w którym komendy MQSC są uruchamiane w zdalnym menedżerze kolejek

Atrybuty obiektów określone w komendach MQSC są wyświetlane w tej sekcji wielkimi literami (na przykład RQMNAME), chociaż nie są one rozróżniane wielkości liter. Nazwy atrybutów komend MQSC są ograniczone do ośmiu znaków.

Komendy MQSC są dostępne na wszystkich platformach. Komendy MQSC są podsumowywane w sekcji Porównywanie zestawów komend.

W systemach Windows, UNIX i Linuxmożna używać komend MQSC jako pojedynczych komend wydawanych w wierszu komend systemu. Aby można było wydać bardziej skomplikowane lub wiele komend, MQSC można utworzyć w pliku, który jest uruchamiany z wiersza komend systemu Windows, UNIX lub Linux . Komendy MQSC mogą być wysyłane do zdalnego menedżera kolejek. Szczegółowe informacje na ten temat zawiera sekcja MQSC reference.

“Komendy skryptowe (MQSC)” na stronie 76 zawiera opis każdej komendy MQSC i jej składni.

Więcej informacji na temat używania komend MQSC w administrowaniu lokalnym zawiera sekcja “Wykonywanie lokalnych zadań administracyjnych za pomocą komend MQSC” na stronie 75 .

Programowalne formaty komend (PCFs)

Programowalne formaty komend (PCFs) definiują komunikaty komend i odpowiedzi, które mogą być wymieniane między programem a dowolnym menedżerem kolejek (który obsługuje systemy PCF) w sieci. Komendy PCF można używać w aplikacji do zarządzania systemami w celu administrowania obiektami produktu IBM WebSphere MQ : obiekty informacji uwierzytelniającej, kanały, obiekty nasłuchiwania kanałów, listy nazw, definicje procesów, menedżery kolejek, kolejki, usługi i klasy pamięci masowej. Aplikacja może działać z jednego punktu w sieci, aby komunikować informacje o komendach i odpowiedziach z dowolnym menedżerem kolejek, lokalnym lub zdalnym za pomocą lokalnego menedżera kolejek.

Więcej informacji na temat systemów PCFs zawiera sekcja “Wprowadzenie do formatów komend programowalnych” na stronie 9.

Definicje PCF i struktur dla komend i odpowiedzi zawiera sekcja Programmable command formats reference(Skorowidz formatów komend programowalnych).

IBM WebSphere MQ Explorer

Za pomocą IBM WebSphere MQ Explorermożna wykonywać następujące czynności:

- Definiowanie i sterowanie różnymi zasobami, w tym menedżerami kolejek, kolejkami, definicjami procesów, listami nazw, kanałami, kanałami połączeń klientów, obiektami nasłuchiwania, usługami i klastrami.
- Uruchamianie lub zatrzymywanie lokalnego menedżera kolejek i powiązanych z nim procesów.
- Wyświetlanie menedżerów kolejek i powiązanych z nimi obiektów na stacji roboczej lub na innych stacjach roboczych.
- Sprawdź status menedżerów kolejek, klastrów i kanałów.
- Sprawdź, które aplikacje, użytkownicy lub kanały mają otwartą określoną kolejkę, z poziomu statusu kolejki.

W systemach Windows i Linux można uruchomić program IBM WebSphere MQ Explorer , korzystając z menu systemowego, pliku wykonywalnego `MQExplorer` lub komendy **`strmqcfg`** .

W systemie Linux, aby pomyślnie uruchomić serwer IBM WebSphere MQ Explorer , należy mieć możliwość zapisu pliku w katalogu osobistym, a katalog osobisty musi istnieć.

Za pomocą produktu IBM WebSphere MQ Explorer można administrować zdalnymi menedżerami kolejek na innych platformach, w tym w produkcie z/OS, w celu uzyskania szczegółowych informacji i pobrania pliku SupportPac MS0T, patrz https://www.ibm.com/support/docview.wss?uid=swg24021041.

Więcej informacji zawiera sekcja “Administrowanie przy użyciu IBM WebSphere MQ Explorer” na stronie 58.

Aplikacja Konfiguracja domyślna produktu Windows

Do utworzenia *uruchamiającego* (lub domyślnego) zestawu obiektów produktu IBM WebSphere MQ można użyć domyślnego programu konfiguracyjnego Windows . Podsumowanie utworzonych obiektów domyślnych znajduje się w tabeli Tabela 1: Obiekty utworzone przez domyślną aplikację konfiguracyjna systemu Windows.

Usługa Microsoft Cluster Service (MSCS)

Usługa Microsoft Cluster Service (MSCS) umożliwia połączenie serwerów z *klastrem*, co zapewnia wyższą dostępność danych i aplikacji, a także ułatwia zarządzanie systemem. MSCS może automatycznie wykrywać i odzyskiwać awarie serwera lub aplikacji.

Ważne jest, aby nie mylić klastrów w sensie MSCS z klastrami produktu IBM WebSphere MQ . Rozróżnienie to:

IBM WebSphere MQKlastry
: są to grupy dwóch lub większej liczby menedżerów kolejek na jednym lub większej liczby komputerów, zapewniające automatyczne połączenie wzajemne i zezwalające na współużytkowanie kolejek między nimi w celu równoważenia obciążenia i nadmiarowości.

Klastry MSCS
: Grupy komputerów, połączone ze sobą i skonfigurowane w taki sposób, że jeśli jedno się nie powiedzie, MSCS wykonuje *przełączenie awaryjne*, przesyłając dane stanu aplikacji z uszkodzonego komputera na inny komputer w klastrze i ponownie inicjując ich działanie.

Obsługa usługi Microsoft Cluster Service (MSCS) zawiera szczegółowe informacje na temat konfigurowania systemu IBM WebSphere MQ dla systemu Windows w celu korzystania z usługi MSCS.

Pojęcia pokrewne

Przegląd techniczny produktu WebSphere MQ

“Administrowanie lokalnymi obiektami IBM WebSphere MQ” na stronie 71
W tej sekcji opisano sposób administrowania lokalnymi obiektami produktu IBM WebSphere MQ w celu obsługi aplikacji, które używają interfejsu kolejki komunikatów (Message Queue Interface-MQI). W tym kontekście lokalne administrowanie oznacza tworzenie, wyświetlanie, zmienianie, kopiowanie i usuwanie obiektów IBM WebSphere MQ .

“Administrowanie zdalnymi obiektami IBM WebSphere MQ” na stronie 108

Uwagi dotyczące utraty kontaktu z menedżerem zasobów XA

Zadania pokrewne

Planowanie

Konfigurowanie

Odsyłacze pokrewne

Scenariusze obsługi transakcyjnej

Administrowanie lokalne i zdalne

Obiekty produktu WebSphere MQ można administrować lokalnie lub zdalnie.

Administrowanie lokalną oznacza wykonywanie zadań administracyjnych na wszystkich menedżerach kolejek, które zostały zdefiniowane w systemie lokalnym. Dostęp do innych systemów można uzyskać, na przykład za pomocą programu emulacji terminalu TCP/IP **telnet**, i przeprowadzając tam administrację. W produkcie WebSphere MQmożna uznać to za administrację lokalną, ponieważ nie są zaangażowane żadne kanały, to znaczy komunikacja jest zarządzana przez system operacyjny.

Produkt WebSphere MQ obsługuje administrowanie z pojedynczego punktu kontaktowego przez to, co jest nazywane *zdalnym administrowaniem*. Umożliwia to wydawanie komend z systemu lokalnego, które

są przetwarzane w innym systemie i ma zastosowanie także do programu WebSphere MQ Explorer. Na przykład można wydać komendę zdalną w celu zmiany definicji kolejki w zdalnym menedżerze kolejek. Użytkownik nie musi logować się do tego systemu, chociaż trzeba mieć zdefiniowane odpowiednie kanały. Musi być uruchomiony menedżer kolejek i serwer komend w systemie docelowym.

Niektóre komendy nie mogą być wydawane w ten sposób, w szczególności w przypadku tworzenia lub uruchamiania menedżerów kolejek i uruchamiania serwerów komend. Aby wykonać ten typ zadania, należy zalogować się do systemu zdalnego i wydać z niego komendy lub utworzyć proces, który będzie mógł wydać komendy. To ograniczenie ma zastosowanie również do programu WebSphere MQ Explorer.

Produkt “Administrowanie zdalnymi obiektami IBM WebSphere MQ” na stronie 108 zawiera szczegółowy opis tematu zdalnego administrowania.

Sposób korzystania z komend sterujących IBM WebSphere MQ

W tej sekcji opisano sposób korzystania z komend sterujących IBM WebSphere MQ .

Jeśli wymagane jest wydanie komend sterujących, ID użytkownika musi należeć do grupy mqm. Więcej informacji na ten temat można znaleźć w sekcji Uprawnienia do administrowania produktem WebSphere MQ w systemach UNIX, Linux i Windows . Ponadto należy zwrócić uwagę na następujące informacje dotyczące środowiska:

WebSphere MQ dla systemu Windows
Wszystkie komendy sterujące mogą być wydawane z poziomu wiersza komend. W nazwach komend i ich flagach nie jest rozróżniana wielkość liter: można wprowadzić je wielkimi literami, małymi lub wielkimi literami. Jednak w argumentach dotyczących komend sterujących (takich jak nazwy kolejek) rozróżniana jest wielkość liter.

W opisach składni łącznik (-) jest używany jako indykator flagi. Zamiast łącznika można użyć ukośnika (/).

WebSphere MQ dla systemów UNIX and Linux
Wszystkie komendy sterujące produktu WebSphere MQ mogą być wydawane z poziomu powłoki. W przypadku wszystkich komend rozróżniana jest wielkość liter.

Podzbiór komend sterujących może zostać wydany za pomocą Eksploratora IBM WebSphere MQ .

Więcej informacji na ten temat zawiera sekcja Komendy sterujące produktu WebSphere MQ .

Automatyzacja zadań administracyjnych

Użytkownik może zdecydować, że będzie on korzystny dla danej instalacji w celu zautomatyzowania niektórych zadań administracyjnych i monitorowania. Można zautomatyzować zadania administracyjne zarówno dla lokalnych, jak i zdalnych menedżerów kolejek za pomocą komend PCF (programmable command format). W tej sekcji założono, że użytkownik ma doświadczenie w administrowaniu obiektami produktu WebSphere MQ .

Komendy PCF

Komendy programu WebSphere MQ programmable command format (PCF) mogą być używane do uruchamiania zadań administracyjnych w programie administracyjnym. W ten sposób z poziomu programu można manipulować obiektami menedżera kolejek (kolejkami, definicjami procesów, listami nazw, kanałami, kanałami połączeń klientów, obiektami nasłuchiwania, usługami i obiektami informacji uwierzytelniających), a nawet samodzielnie operować menedżerami kolejek.

Komendy PCF obejmują ten sam zakres funkcji udostępnianych przez komendy MQSC. Można napisać program, aby wydać komendy PCF do dowolnego menedżera kolejek w sieci z jednego węzła. W ten sposób można zarówno scentralizować, jak i automatyzować zadania administracyjne.

Każda komenda PCF jest strukturą danych osadzoną w części danych aplikacji komunikatu produktu WebSphere MQ . Każda komenda jest wysyłana do docelowego menedżera kolejek za pomocą funkcji MQI MQPUT w taki sam sposób, jak w przypadku innych komunikatów. Jeśli serwer komend jest uruchomiony w menedżerze kolejek odbierającym komunikat, serwer komend interpretuje go jako komunikat komendy

i uruchamia komendę. Aby uzyskać odpowiedzi, aplikacja wysyła wywołanie MQGET , a dane odpowiedzi są zwracane w innej strukturze danych. Aplikacja może następnie przetworzyć odpowiedź i podjąć odpowiednie działania.

Uwaga: W przeciwieństwie do komend MQSC, komendy PCF i ich odpowiedzi nie są w formacie tekstowym, który można odczytać.

Krótko mówiąc, są to niektóre z elementów potrzebnych do utworzenia komunikatu komendy PCF:

deskryptor komunikatu
Jest to standardowy deskryptor komunikatu produktu WebSphere MQ , w którym:

- Typ komunikatu (*MsqType*) to MQMT_REQUEST.
- Format komunikatu (*Format*) to MQFMT_ADMIN.

Dane aplikacji
Zawiera komunikat PCF zawierający nagłówek PCF, w którym:

- Typ komunikatu PCF (*Type*) określa komendę MQCFT_COMMAND.
- Identyfikator komendy określa komendę, na przykład *Change Queue* (MQCMD_CHANGE_Q).

Pełny opis struktur danych PCF i sposób ich implementowania zawiera sekcja "Wprowadzenie do formatów komend programowalnych" na stronie 9.

Atrybuty obiektu PCF

Atrybuty obiektów w programie PCF nie są ograniczone do ośmiu znaków, ponieważ są przeznaczone dla komend MQSC. Są one przedstawione w tym przewodniku kursywą. Na przykład odpowiednikiem PCF w produkcie RQMNAME jest *RemoteQMgrName*.

Zmiany w systemach PCF

Zmiany PCF to komendy PCF, które zawierają komendy MQSC w tekście komunikatu. Komend PCF można używać do wysyłania komend do zdalnego menedżera kolejek. Więcej informacji na temat systemów PCF o zmienionym znaczeniu zawiera sekcja Escape(Escape).

Wprowadzenie do formatów komend programowalnych

Programowalne formaty komend (PCFs) definiują komunikaty komend i odpowiedzi, które mogą być wymieniane między programem a dowolnym menedżerem kolejek (który obsługuje systemy PCF) w sieci. PCF upraszczają administrację menedżera kolejek i inne administrowanie siecią. Mogą one być wykorzystywane do rozwiązywania problemu skomplikowanej administracji rozproszonych sieci, zwłaszcza, gdy sieci rosną w wielkości i złożoności.

Formaty komend programowalnych opisane w tej dokumentacji produktu są obsługiwane przez:

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

Problem z komendami PCF

Administrowanie sieciami rozproszonymi może stać się skomplikowane. Problemy administracji w dalszym ciągu rosną w miarę wzrostu i złożoności sieci.

Przykłady administrowania specyficzne dla przesyłania komunikatów i kolejkowania obejmują:

- Zarządzanie zasobami.

Na przykład tworzenie kolejek i usuwanie.

- Monitorowanie wydajności.

  Na przykład: maksymalna głębokość kolejki lub szybkość komunikatów.

- .

  Na przykład: strojenie parametrów kolejki, takich jak maksymalna głębokość kolejki, maksymalna długość komunikatu oraz włączanie i wyłączanie kolejek.

- Kierowanie komunikatów.

  Definicja tras alternatywnych poprzez sieć.

Komendy PCF produktu WebSphere MQ mogą być używane do uproszczenia administrowania menedżerem kolejek i innej administracji sieciowej. Komendy PCF umożliwiają korzystanie z jednej aplikacji w celu wykonania administrowania siecią z jednego menedżera kolejek w sieci.

Co to są PCFs?

Systemy PCF definiują komunikaty komend i odpowiedzi, które mogą być wymieniane między programem a dowolnym menedżerem kolejek (który obsługuje systemy PCF) w sieci. Komendy PCF można używać w programie aplikacji do zarządzania systemami w celu administrowania obiektami WebSphere MQ : obiektami informacji uwierzytelniających, kanałami, obiektami nasłuchiwania kanałów, listami nazw, definicjami procesów, menedżerami kolejek, kolejkami, usługami i klasami pamięci masowej. Aplikacja może działać z jednego punktu w sieci, aby komunikować informacje o komendach i odpowiedziach z dowolnym menedżerem kolejek, lokalnym lub zdalnym za pomocą lokalnego menedżera kolejek.

Każdy menedżer kolejek ma kolejkę administracyjną o standardowej nazwie kolejki, a aplikacja może wysyłać komunikaty komend PCF do tej kolejki. Każdy menedżer kolejek posiada również serwer komend do obsługi komunikatów komend z kolejki administracyjnej. Komunikaty komend PCF mogą więc być przetwarzane przez dowolny menedżer kolejek w sieci, a dane odpowiedzi mogą być zwracane do aplikacji przy użyciu określonej kolejki odpowiedzi. Komendy PCF i komunikaty odpowiedzi są wysyłane i odbierane przy użyciu normalnego interfejsu kolejki komunikatów (Message Queue Interface-MQI).

Listę dostępnych komend PCF, w tym ich parametrów, można znaleźć w sekcji Definicje formatów komend programowalnych.

Korzystanie z formatów komend programowalnych

W programie WebSphere MQ zdalnej administracji można używać systemów PCF w programie do zarządzania systemami.

W tej sekcji zawarto następujące informacje:

- “Komunikaty komend PCF” na stronie 10
- “Odpowiedzi” na stronie 13
- Reguły dotyczące nazewnictwa obiektów produktu IBM WebSphere MQ
- “Sprawdzanie uprawnień dla komend PCF” na stronie 15

Komunikaty komend PCF

Komunikaty komend PCF składają się z nagłówka PCF, parametrów zidentyfikowanych w tym nagłówku oraz danych komunikatu zdefiniowanych przez użytkownika. Komunikaty są wysyłane za pomocą wywołań interfejsu kolejki komunikatów.

Każda komenda i jej parametry są wysyłane jako osobny komunikat komendy zawierający nagłówek PCF, po którym następuje pewna liczba struktur parametrów. Szczegółowe informacje na temat nagłówka PCF można znaleźć w sekcji MQCFH-PCF header, a na przykład w strukturze parametrów można znaleźć w sekcji MQCFST-PCF string parameter(MQCFST-parametr łańcucha PCF). Nagłówek PCF identyfikuje komendę i liczbę struktur parametrów, które są zgodne w tym samym komunikacie. Każda struktura parametru udostępnia parametr komendy.

Odpowiedzi na komendy wygenerowane przez serwer komend mają podobną strukturę. Istnieje nagłówek PCF, po którym następuje liczba struktur parametrów. Odpowiedzi mogą składać się z więcej niż jednego komunikatu, ale komendy zawsze składają się tylko z jednego komunikatu.

Na platformach innych niż z/OSkolejka, do której wysyłane są komendy PCF, jest zawsze nazywana SYSTEM.ADMIN.COMMAND.QUEUE.

Jak wystawiać komunikaty komend PCF

Użyj zwykłych wywołań interfejsu MQI (Message Queue Interface), MQPUT, MQGET i tak dalej, aby umieścić i pobrać komunikaty komend i odpowiedzi PCF do kolejek i ich z nich.

Uwaga:

Upewnij się, że serwer komend jest uruchomiony w docelowym menedżerze kolejek dla komendy PCF, która ma zostać wykonana w tym menedżerze kolejek.

Lista dostarczonych plików nagłówkowych znajduje się w sekcji WebSphere MQ COPY, header, include and module files(Pliki nagłówkowe, nagłówki i pliki modułów).

Deskryptor komunikatu dla komendy PCF

Deskryptor komunikatu produktu WebSphere MQ jest w pełni udokumentowany w sekcji MQMD-deskryptor komunikatu.

Komunikat komendy PCF zawiera następujące pola w deskryptorze komunikatu:

Report
Poprawna wartość, jeśli jest to wymagane.

MsgType
To pole musi mieć wartość MQMT_REQUEST, aby wskazać komunikat wymagający odpowiedzi.

Expiry
Poprawna wartość, jeśli jest to wymagane.

Feedback
Ustaw na wartość MQFB_NONE

Encoding
W przypadku wysyłania do systemów Windows, UNIX lub Linux należy ustawić to pole na kodowanie używane w przypadku danych komunikatu. W razie potrzeby konwersja jest wykonywana.

CodedCharSetId
Jeśli wysyłasz doW systemach Windows, UNIX lub Linux należy ustawić to pole na identyfikator kodowanego zestawu znaków używany dla danych komunikatu. W razie potrzeby konwersja jest wykonywana.

Format
Ustaw wartość MQFMT_ADMIN.

Priority
Poprawna wartość, jeśli jest to wymagane.

Persistence
Poprawna wartość, jeśli jest to wymagane.

MsgId
Aplikacja wysyłający może określić dowolną wartość lub można określić wartość MQMI_NONE w celu żądania menedżera kolejek w celu wygenerowania unikalnego identyfikatora komunikatu.

CorrelId
Aplikacja wysyłający może określić dowolną wartość, a wartość MQCI_NONE można określić, aby wskazać brak identyfikatora korelacji.

ReplyToQ
Nazwa kolejki, w której ma zostać odebranie odpowiedzi.

ReplyToQMgr
Nazwa menedżera kolejek dla odpowiedzi (lub pusta).

Pola kontekstu komunikatu
W razie potrzeby te pola mogą być ustawione na dowolne poprawne wartości. Zwykle opcja komunikatu umieszczanie MQPMO_DEFAULT_CONTEXT jest używana do ustawiania pól kontekstu komunikatu na wartości domyślne.

Jeśli używana jest struktura MQMD w wersji version-2 , należy ustawić następujące dodatkowe pola:

GroupId
Ustaw na wartość MQGI_NONE

MsgSeqNumber
Ustaw na 1

Offset
Ustaw na 0

MsgFlags
Ustaw na MQMF_NONE

OriginalLength
Ustaw na wartość MQOL_UNDEFINED

Wysyłanie danych użytkownika

Struktury PCF mogą być również używane do wysyłania danych komunikatów zdefiniowanych przez użytkownika. W takim przypadku pole deskryptora komunikatu *Format* musi być ustawione na wartość MQFMT_PCF.

Wysyłanie i odbieranie komunikatów PCF w określonej kolejce

Wysyłanie komunikatów PCF do określonej kolejki

Aby wysłać komunikat do określonej kolejki, wywołanie komendy mqPutBag przekształca treść określonej torby w komunikat PCF i wysyła komunikat do określonej kolejki. Zawartość worka pozostaje bez zmian po wywołaniu.

Jako dane wejściowe dla tego wywołania należy podać:

- Uchwyt połączenia MQI.
- Uchwyt obiektu dla kolejki, w której ma być umieszczony komunikat.
- Deskryptor komunikatu. Więcej informacji na temat deskryptora komunikatu zawiera sekcja MQMD-deskryptor komunikatu.
- Umieść opcje komunikatów przy użyciu struktury MQPMO. Więcej informacji na temat struktury MQPMO zawiera sekcja Opcje komunikatu MQPMO-Put-message.
- Uchwyt worka, który ma zostać przekształcony w komunikat.

  Uwaga: Jeśli w torbie znajduje się komunikat administracyjny, a do wstawiania wartości do worka użyto wywołania mqAddInquiry, wartość elementu danych MQIASY_COMMAND musi być komendą INQUIRE rozpoznawaną przez interfejs MQAI.

Pełny opis wywołania funkcji mqPutBag znajduje się w sekcji mqPutBag(Wtorka dla produktu MQ).

Odbieranie komunikatów PCF z określonej kolejki

Aby odebrać komunikat z określonej kolejki, wywołanie funkcji Bag mqGetpobiera komunikat PCF z określonej kolejki i przekształca dane komunikatu w worek danych.

Jako dane wejściowe dla tego wywołania należy podać:

- Uchwyt połączenia MQI.
- Uchwyt obiektu kolejki, z której ma być odczytany komunikat.
- Deskryptor komunikatu. W strukturze MQMD parametr `Format` musi mieć wartość MQFMT_ADMIN, MQFMT_EVENT, lub MQFMT_PCF.

  Uwaga: Jeśli komunikat jest odbierany w ramach jednostki pracy (czyli z opcją MQGMO_SYNCPOINT), a komunikat ma nieobsługiwany format, to można utworzyć kopię zapasową jednostki pracy. Następnie komunikat zostanie przywrócony do kolejki i można go pobrać za pomocą wywołania MQGET zamiast wywołania komendy mqGet. Więcej informacji na temat deskryptora komunikatu można znaleźć w sekcji Opcje MQGMO-Get-message.
- Pobierz opcje komunikatu przy użyciu struktury MQGMO. Więcej informacji na temat struktury MQGMO zawiera sekcja MQMD-deskryptor komunikatu.
- Uchwyt worka, który ma zawierać przekształcony komunikat.

Pełny opis wywołania funkcji mqGetBag znajduje się w sekcji mqGetBag(Wtorek mqGet).

Odpowiedzi

W odpowiedzi na każdą komendę serwer komend generuje jeden lub większą liczbę komunikatów odpowiedzi. Komunikat odpowiedzi ma podobny format do komunikatu komendy.

Nagłówek PCF ma taką samą wartość identyfikatora komendy, jak komenda, do której jest ona odpowiedzią (szczegółowe informacje zawiera sekcja MQCFH-PCF header). Identyfikator komunikatu i identyfikator korelacji są ustawiane zgodnie z opcjami raportu dla żądania.

Jeśli typem nagłówka PCF komunikatu komendy jest MQCFT_COMMAND, generowane są tylko standardowe odpowiedzi. Takie komendy są obsługiwane na wszystkich platformach z wyjątkiem systemu z/OS. Starsze aplikacje nie obsługują PCF w systemie z/OS; w programie WebSphere MQ Windows Explorer jest to jedna taka aplikacja (jednak wersja 6.0 lub nowsza IBM WebSphere MQ Explorer obsługuje PCF w systemie z/OS).

Jeśli typem nagłówka PCF komunikatu komendy jest MQCFT_COMMAND_XR, wówczas generowane są rozszerzone lub standardowe odpowiedzi. Takie komendy są obsługiwane w systemie z/OS i na niektórych innych platformach. Komendy wydawane w systemie z/OS generują tylko rozszerzone odpowiedzi. Na innych platformach może zostać wygenerowany dowolny typ odpowiedzi.

Jeśli pojedyncza komenda określa ogólną nazwę obiektu, w komunikacie dla każdego zgodnego obiektu zwracana jest osobna odpowiedź. W przypadku generowania odpowiedzi pojedyncza komenda z nazwą ogólną jest traktowana jako wiele pojedynczych komend (z wyjątkiem pola sterującego MQCFC_LAST lub MQCFC_NOT_LAST). W przeciwnym razie jeden komunikat komendy generuje jeden komunikat odpowiedzi.

Niektóre odpowiedzi PCF mogą zwrócić strukturę nawet wtedy, gdy nie jest ona żądana. Struktura ta jest wyświetlana w definicji odpowiedzi (Definicje formatów komend programowalnych) jako *zawsze zwracanej*. Przyczyna, dla której w przypadku tych odpowiedzi należy podać nazwę obiektów w odpowiedzi, aby zidentyfikować obiekt, którego dotyczą dane.

Deskryptor komunikatu dla odpowiedzi

W komunikacie odpowiedzi znajdują się następujące pola w deskryptorze komunikatu:

MsgType
To pole ma wartość MQMT_REPLY.

MsgId
To pole jest generowane przez menedżer kolejek.

CorrelId
To pole jest generowane zgodnie z opcjami raportu w komunikacie komendy.

Format
To pole to MQFMT_ADMIN.

Encoding
Ustaw wartość MQENC_NATIVE.

CodedCharSetId
Ustaw wartość MQCCSI_Q_MGR.

Persistence
To samo, co w komunikacie komendy.

Priority
To samo, co w komunikacie komendy.

Odpowiedź jest generowana z opcją MQPMO_PASS_IDENTITY_CONTEXT.

Odpowiedzi standardowe

Komunikaty komend z typem nagłówka MQCFT_COMMAND, generowane są standardowe odpowiedzi. Takie komendy są obsługiwane na wszystkich platformach z wyjątkiem systemu z/OS.

Istnieją trzy typy standardowej odpowiedzi:

- Odpowiedź OK
- Odpowiedź na błąd
- Odpowiedź na dane

Odpowiedź OK

Ta odpowiedź składa się z komunikatu rozpoczynający się od nagłówka formatu komend, w polu *CompCode* (MQCC_OK) lub MQCC_WARNING (MQCC_WARNING).

W przypadku wywołania MQCC_OK *Reason* ma wartość MQRC_NONE.

W przypadku wartości MQCC_WARNING *Reason* identyfikuje rodzaj ostrzeżenia. W tym przypadku po nagłówku formatu komendy może następować jedna lub więcej struktur parametrów ostrzegawczych, które są odpowiednie dla tego kodu przyczyny.

W obu przypadkach, w przypadku komendy inquire, dalsze struktury parametrów mogą być zgodne z opisem w poniższych sekcjach.

Odpowiedź na błąd

Jeśli komenda ma błąd, wysyłany jest jeden lub więcej komunikatów o odpowiedzi na błędy (więcej niż jeden może zostać wysłany nawet dla komendy, która normalnie ma tylko jeden komunikat odpowiedzi). Komunikaty odpowiedzi o błędach mają odpowiednio ustawione MQCFC_LAST lub MQCFC_NOT_LAST.

Każdy taki komunikat rozpoczyna się od nagłówka w formacie odpowiedzi, którego wartość *CompCode* ma wartość MQCC_FAILED i pole *Reason* , które identyfikuje konkretny błąd. W ogólnym przypadku każdy komunikat opisuje inny błąd. Ponadto każdy komunikat ma zero lub jeden (nigdy więcej niż jeden) struktury parametrów błędu po nagłówku. Ta struktura parametrów, jeśli istnieje, jest strukturą MQCFIN, z polem *Parameter* zawierającym jedną z następujących wartości:

- MQIACF_PARAMETER_ID,

  Pole *Value* w strukturze jest identyfikatorem parametru, który wystąpił w błędzie (na przykład MQCA_Q_NAME).
- MQIACF_ERROR_ID

  Ta wartość jest używana z wartością *Reason* (w nagłówku formatu komendy) błędu MQRC_UNEXPECTED_ERROR. Pole *Value* w strukturze MQCFIN jest nieoczekiwanym kodem przyczyny odebranym przez serwer komend.
- MQIACF_SELECTOR

  Ta wartość występuje wtedy, gdy struktura listy (MQCFIL) wysłana z komendą zawiera zduplikowany selektor lub jeden, który jest niepoprawny. Pole *Reason* w nagłówku formatu komendy identyfikuje

błąd, a pole *Value* w strukturze MQCFIN jest wartością parametru w strukturze MQCFIL komendy, która była błędna.

- MQIACF_ERROR_OFFSET

  Ta wartość występuje wtedy, gdy w komendzie Ping Channel wystąpił błąd porównywania danych. Pole *Value* w strukturze jest przesuniętym błędem porównania kanału Ping.

- MQIA_CODED_CHAR_SET_ID

  Ta wartość występuje wtedy, gdy identyfikator kodowanego zestawu znaków w deskryptorze komunikatu przychodzącego komunikatu komendy PCF nie jest zgodny z identyfikatorem docelowego menedżera kolejek. Pole *Value* w strukturze jest kodowanym identyfikatorem zestawu znaków menedżera kolejek.

Ostatni (lub tylko) komunikat odpowiedzi o błędzie jest odpowiedzią podsumowową z polem *CompCode* o wartości MQCC_FAILED i polem *Reason* o nazwie MQRCCF_COMMAND_FAILED. Ten komunikat nie ma struktury parametru po nagłówku.

Odpowiedź na dane

Ta odpowiedź składa się z odpowiedzi OK (zgodnie z wcześniejszym opisem) do komendy inquire. Po kliknięciu przycisku OK zostaną wykonane dodatkowe struktury zawierające żądane dane zgodnie z opisem w sekcji Definicje formatów komend programowalnych.

Aplikacje nie mogą zależeć od tych dodatkowych struktur parametrów, które są zwracane w dowolnej kolejności.

Sprawdzanie uprawnień dla komend PCF

Po przetworzeniu komendy PCF *UserIdentifier* z deskryptora komunikatu w komunikacie komendy jest używany dla wymaganych sprawdzeń uprawnień do obiektu produktu WebSphere MQ . Sprawdzanie uprawnień jest implementowane w inny sposób na każdej platformie zgodnie z opisem w tym temacie.

Sprawdzenia są wykonywane w systemie, w którym komenda jest przetwarzana; dlatego ten identyfikator użytkownika musi istnieć w systemie docelowym i musi mieć wymagane uprawnienia do przetwarzania komendy. Jeśli komunikat pochodzi z systemu zdalnego, jednym ze sposobów na osiągnięcie identyfikatora istniejącego w systemie docelowym jest posiadanie zgodnego identyfikatora użytkownika zarówno w systemie lokalnym, jak i zdalnym.

IBM WebSphere MQ dla systemów Windows, UNIX and Linux

Windows UNIX Linux

Aby możliwe było przetworzenie dowolnej komendy PCF, identyfikator użytkownika musi mieć uprawnienie *dsp* dla obiektu menedżera kolejek w systemie docelowym. Ponadto w przypadku niektórych komend PCF wykonywane są operacje sprawdzania uprawnień do obiektu WebSphere MQ , jak to pokazano na rysunku Tabela 1 na stronie 16.

Aby przetworzyć jedną z następujących komend , identyfikator użytkownika musi należeć do grupy *mqm*.

Uwaga: W przypadku systemu Windows **tylko**identyfikator użytkownika może należeć do grupy *Administratorzy* lub grupy *mqm*.

- Zmień kanał
- Kopiuj kanał
- Utwórz kanał
- Usuń kanał
- Kanał ping
- Resetowanie kanału
- Uruchom kanał

- Zamknij kanał
- Uruchom inicjator kanału
- Uruchom program nasłuchujący kanału
- Rozstrzyganie kanału
- Resetowanie klastra
- Odśwież klaster
- Menedżer kolejki - SUSPEND
- Wznów menedżer kolejek

WebSphere MQ dla produktu HP Integrity NonStop Server

Aby możliwe było przetworzenie dowolnej komendy PCF, identyfikator użytkownika musi mieć uprawnienie *dsp* dla obiektu menedżera kolejek w systemie docelowym. Dodatkowo, dla niektórych komend PCF wykonywane są sprawdzenia uprawnień do obiektu IBM WebSphere MQ , jak to pokazano na Tabela 1 na stronie 16.

Aby przetworzyć dowolne z następujących komend ID użytkownika musi należeć do grupy *mqm*:

- Zmień kanał
- Kopiuj kanał
- Utwórz kanał
- Usuń kanał
- Kanał ping
- Resetowanie kanału
- Uruchom kanał
- Zamknij kanał
- Uruchom inicjator kanału
- Uruchom program nasłuchujący kanału
- Rozstrzyganie kanału
- Resetowanie klastra
- Odśwież klaster
- Menedżer kolejki - SUSPEND
- Wznów menedżer kolejek

Uprawnienia do obiektu WebSphere MQ

Tabela 1. Systemy Windows, HP Integrity NonStop Server, UNIX and Linux -uprawnienia do obiektów

Komenda	**Uprawnienie do obiektu WebSphere MQ**	**Uprawnienie klasy (dla typu obiektu)**
Zmień informacje uwierzytelniające	dsp i chg	nie dotyczy
Zmień kanał	dsp i chg	nie dotyczy
Zmiana obiektu nasłuchiwania kanału	dsp i chg	nie dotyczy
Zmień kanał połączenia klienta	dsp i chg	nie dotyczy
Zmień listę nazw	dsp i chg	nie dotyczy
Zmień proces	dsp i chg	nie dotyczy

Tabela 1. Systemy Windows, HP Integrity NonStop Server, UNIX and Linux -uprawnienia do obiektów (kontynuacja)

Komenda	**Uprawnienie do obiektu WebSphere MQ**	**Uprawnienie klasy (dla typu obiektu)**
Zmiana kolejki	dsp i chg	nie dotyczy
Zmiana menedżera kolejek	chg *patrz uwaga 3 i uwaga 5*	nie dotyczy
Zmień usługę	dsp i chg	nie dotyczy
Wyczyść kolejkę	clr	nie dotyczy
Kopiowanie informacji uwierzytelniających	dsp	crt
Kopiowanie informacji uwierzytelniających (zastępowanie) *patrz uwaga 1*	*from:* dsp *to:* chg	crt
Kopiuj kanał	dsp	crt
Kopiuj kanał (zastępowanie) *patrz uwaga 1*	*from:* dsp *to:* chg	crt
Kopiuj program nasłuchujący kanału	dsp	crt
Kopiowanie procesu nasłuchiwania kanału (zastępowanie) *patrz uwaga 1*	*from:* dsp *to:* chg	crt
Kopiuj kanał połączenia klienta	dsp	crt
Kopiuj kanał połączenia klienta (zastępowanie) *patrz uwaga 1*	*from:* dsp *to:* chg	crt
Kopiuj listę nazw	dsp	crt
Kopiowanie listy nazw (Replace) *patrz Uwaga 1*	*from:* dsp *to:* dsp i chg	crt
Kopiuj proces	dsp	crt
Kopiowanie procesu (zastępowanie) *patrz uwaga 1*	*from:* dsp *to:* chg	crt
Kopiuj kolejkę	dsp	crt
Kopiowanie kolejki (zastępowanie) *patrz uwaga 1*	*from:* dsp *to:* dsp i chg	crt
Tworzenie informacji uwierzytelniających	*(domyślne informacje o uwierzytelnianiu systemu)* dsp	crt
Tworzenie informacji uwierzytelniających (zastępowanie) *patrz uwaga 1*	*(domyślne informacje o uwierzytelnianiu systemu)* dsp *to:* chg	crt
Utwórz kanał	*(domyślny kanał systemowy)* dsp	crt
Tworzenie kanału (zastępowanie) *patrz uwaga 1*	*(domyślny kanał systemowy)* dsp *to:* chg	crt
Tworzenie nasłuchiwania kanału	*(domyślny program nasłuchujący systemu)* dsp	crt

Tabela 1. Systemy Windows, HP Integrity NonStop Server, UNIX and Linux -uprawnienia do obiektów (kontynuacja)		
Komenda	**Uprawnienie do obiektu WebSphere MQ**	**Uprawnienie klasy (dla typu obiektu)**
Tworzenie nasłuchiwania kanału (zastępowanie) *patrz uwaga 1*	*(domyślny program nasłuchujący systemu)* dsp *to:* chg	crt
Utwórz kanał połączenia klienta	*(domyślny kanał systemowy)* dsp	crt
Utwórz kanał połączenia klienta (zastępowanie) *patrz uwaga 1*	*(domyślny kanał systemowy)* dsp *to:* chg	crt
Utwórz listę nazw	*(systemowa domyślna lista nazw)* dsp	crt
Tworzenie listy nazw (Replace) *patrz Uwaga 1*	*(systemowa domyślna lista nazw)* dsp *to:* dsp i chg	crt
Utwórz proces	*(domyślny proces systemowy)* dsp	crt
Utwórz proces (zastępowanie) *patrz uwaga 1*	*(domyślny proces systemowy)* dsp *to:* chg	crt
Tworzenie kolejki	*(systemowa kolejka domyślna)* dsp	crt
Tworzenie kolejki (zastępowanie) *patrz uwaga 1*	*(systemowa kolejka domyślna)* dsp *to:* dsp i chg	crt
Utwórz usługę	*(systemowa kolejka domyślna)* dsp	crt
Utwórz usługę (zastępowanie) *patrz uwaga 1*	*(systemowa kolejka domyślna)* dsp *to:* chg	crt
Usuń informacje uwierzytelniające	dsp i dlt	nie dotyczy
Usuń rekord uprawnień	*(obiekt menedżera kolejek)* chg *patrz Uwaga 4*	*patrz Uwaga 4*
Usuń kanał	dsp i dlt	nie dotyczy
Usuń program nasłuchujący kanału	dsp i dlt	nie dotyczy
Usuń kanał połączenia klienta	dsp i dlt	nie dotyczy
Usuń listę nazw	dsp i dlt	nie dotyczy
Usuń proces	dsp i dlt	nie dotyczy
Usuń kolejkę	dsp i dlt	nie dotyczy
Usuwanie usługi	dsp i dlt	nie dotyczy
Sprawdzanie informacji uwierzytelniających	dsp	nie dotyczy
Sprawdź rekordy uprawnień	*patrz Uwaga 4*	*patrz Uwaga 4*
Sprawdź kanał	dsp	nie dotyczy
Sprawdź program nasłuchujący kanału	dsp	nie dotyczy
Sprawdzanie statusu kanału (dla **ChannelType** MQCHT_CLSSDR)	inq	nie dotyczy

Tabela 1. Systemy Windows, HP Integrity NonStop Server, UNIX and Linux -uprawnienia do obiektów (kontynuacja)		
Komenda	**Uprawnienie do obiektu WebSphere MQ**	**Uprawnienie klasy (dla typu obiektu)**
Sprawdź kanał połączenia klienta	dsp	nie dotyczy
Sprawdź listę nazw	dsp	nie dotyczy
Sprawdź proces	dsp	nie dotyczy
Sprawdź kolejkę	dsp	nie dotyczy
Zapytaj menedżera kolejek	*patrz uwaga 3*	nie dotyczy
Sprawdź status kolejki	dsp	nie dotyczy
Sprawdź usługę	dsp	nie dotyczy
Kanał ping	ctrl	nie dotyczy
Menedżer kolejek ping	*patrz uwaga 3*	nie dotyczy
Odśwież menedżera kolejek	(obiekt menedżera kolejek) chg	nie dotyczy
Odśwież zabezpieczenia (dla **SecurityType** MQSECTYPE_SSL)	(obiekt menedżera kolejek) chg	nie dotyczy
Resetowanie kanału	ctrlx	nie dotyczy
Resetowanie menedżera kolejek	(obiekt menedżera kolejek) chg	nie dotyczy
Resetuj statystyki kolejki	dsp i chg	nie dotyczy
Rozstrzyganie kanału	ctrlx	nie dotyczy
Ustaw rekord uprawnień	*(obiekt menedżera kolejek)* chg *patrz Uwaga 4*	*patrz Uwaga 4*
Uruchom kanał	ctrl	nie dotyczy
Zamknij kanał	ctrl	nie dotyczy
Zatrzymaj połączenie	(obiekt menedżera kolejek) chg	nie dotyczy
Uruchom proces nasłuchujący	ctrl	nie dotyczy
Zatrzymaj proces nasłuchujący	ctrl	nie dotyczy
Uruchom usługę	ctrl	nie dotyczy
Zatrzymaj usługę	ctrl	nie dotyczy
Esc	*patrz Uwaga 2*	*patrz Uwaga 2*

Uwagi:

1. Ta komenda ma zastosowanie, jeśli obiekt, który ma zostać zastąpiony, istnieje, w przeciwnym razie kontrola uprawnień jest taka, jak dla operacji Create (tworzenie) lub Copy bez zastępowania (Replace).
2. Wymagane uprawnienia są określane przez komendę MQSC zdefiniowaną przez tekst zmiany znaczenia, która jest równoważna jednej z poprzednich komend.
3. Aby możliwe było przetworzenie dowolnej komendy PCF, identyfikator użytkownika musi mieć uprawnienie dsp dla obiektu menedżera kolejek w systemie docelowym.

4. Ta komenda PCF jest autoryzowana, jeśli serwer komend nie został uruchomiony z parametrem -a. Domyślnie serwer komend jest uruchamiany, gdy menedżer kolejek jest uruchomiony i bez parametru -a. Więcej informacji na ten temat można znaleźć w Podręczniku administrowania systemem.
5. Nadawanie ID użytkownika *chg* uprawnień dla menedżera kolejek daje możliwość ustawiania rekordów uprawnień dla wszystkich grup i użytkowników. Nie należy przyznawać tego uprawnienia zwykłym użytkownikom lub aplikacjom.

Produkt WebSphere MQ udostępnia również niektóre punkty wyjścia zabezpieczeń kanału, dzięki czemu można będzie dostarczać własne programy obsługi wyjścia dla sprawdzania zabezpieczeń. Szczegółowe informacje można znaleźć w podręczniku Wyświetlanie kanału .

Korzystanie z interfejsu MQAI w celu uproszczenia korzystania z systemów PCF

Interfejs MQAI jest interfejsem administracyjnym dla produktu WebSphere MQ , który jest dostępny na platformach AIX, HP-UX, IBM i, Linux, Solaris i Windows .

Interfejs MQAI wykonuje zadania administracyjne w menedżerze kolejek przy użyciu *worków danych*. Worki danych pozwalają na obsługę właściwości (lub parametrów) obiektów w sposób łatwiejszy niż przy użyciu PCFs.

Użyj interfejsu MQAI w następujący sposób:

Aby uprościć korzystanie z komunikatów PCF
Interfejs MQAI to łatwy sposób administrowania produktem WebSphere MQ. Nie jest konieczne zapisywanie własnych komunikatów PCF, co pozwala uniknąć problemów związanych ze złożonymi strukturami danych.

Aby przekazać parametry w programach napisanych za pomocą wywołań MQI, komunikat PCF musi zawierać komendę oraz szczegółowe informacje na temat danych łańcucha lub liczb całkowitych. W tym celu potrzebne jest kilka instrukcji w programie dla każdej struktury, a przestrzeń pamięci musi być przydzielona. To zadanie może być długie i pracochłonne.

Programy napisane przy użyciu parametrów MQAI przechodzą do odpowiedniego worka danych i potrzebne jest tylko jedno oświadczenie dla każdej struktury. Użycie toreb danych MQAI usuwa potrzebę obsługi tablic i przydzielania pamięci masowej, a także zapewnia pewien stopień izolacji od szczegółów PCF.

Aby łatwiej było obsłużyć warunki błędów
Trudno jest uzyskać kody powrotu z komend PCF, ale interfejs MQAI ułatwia obsługę warunków błędów przez program.

Po utworzeniu i zapełnieniu worka danych można wysłać komunikat komendy administracyjnej do serwera komend menedżera kolejek przy użyciu wywołania `mqExecute` , które oczekuje na komunikaty odpowiedzi. Wywołanie `mqExecute` obsługuje wymianę z serwerem komend i zwraca odpowiedzi w *worku odpowiedzi*.

Więcej informacji na temat interfejsu MQAI zawiera sekcja “Wprowadzenie do interfejsu administracyjnego produktu IBM WebSphere MQ (MQAI)” na stronie 20.

Wprowadzenie do interfejsu administracyjnego produktu IBM WebSphere MQ (MQAI)

Interfejs administracyjny produktu IBM WebSphere MQ (MQAI) jest interfejsem programistycznym dla produktu IBM WebSphere MQ. Wykonuje zadania administracyjne w menedżerze kolejek produktu IBM WebSphere MQ przy użyciu worków danych w celu obsługi właściwości (lub parametrów) obiektów w sposób łatwiejszy niż przy użyciu formatów komend programowalnych (Programmable Command Formats-PCFs).

Pojęcia i terminologia MQAI

Interfejs MQAI to interfejs programistyczny produktu WebSphere MQ, który używa języka C, a także języka Visual Basic dla systemu Windows. Jest ona dostępna na platformach innych niż z/OS.

Wykonuje zadania administracyjne w menedżerze kolejek produktu WebSphere MQ przy użyciu worków danych. Worki danych umożliwiają obsługę właściwości (lub parametrów) obiektów w sposób łatwiejszy niż przy użyciu innego interfejsu administracyjnego, programowalnych formatów komend (PCFs). Interfejs MQAI ułatwia manipulowanie systemami PCFs niż wywołania MQGET i MQPUT.

Więcej informacji na temat toreb danych zawiera sekcja "Torby danych" na stronie 48. Więcej informacji na temat systemów PCFs zawiera sekcja "Wprowadzenie do formatów komend programowalnych" na stronie 9 .

Korzystanie z interfejsu MQAI

Istnieje możliwość użycia interfejsu MQAI do:

- Uprość korzystanie z komunikatów PCF. Interfejs MQAI to łatwy sposób administrowania produktem WebSphere MQ. Nie jest konieczne zapisywanie własnych komunikatów PCF, a tym samym uniknięcie problemów związanych ze złożonymi strukturami danych.
- Obsługa warunków błędów jest łatwiejsza. Trudno jest uzyskać kody powrotu z komend skryptowych WebSphere MQ (MQSC), ale interfejs MQAI ułatwia obsługę warunków błędów przez program.
- Wymiana danych między aplikacjami. Dane aplikacji są wysyłane w formacie PCF, a następnie spakowane i rozpakowane przez interfejs MQAI. Jeśli dane komunikatu składają się z liczb całkowitych i łańcuchów znaków, można użyć interfejsu MQAI, aby skorzystać z wbudowanej konwersji danych produktu WebSphere MQ na potrzeby danych PCF. Pozwala to uniknąć konieczności pisania wyjść konwersji danych. Więcej informacji na temat używania interfejsu MQAI do administrowania produktem WebSphere MQ oraz do wymiany danych między aplikacjami zawiera sekcja "Korzystanie z interfejsu MQAI w celu uproszczenia korzystania z systemów PCF" na stronie 20.

Przykłady użycia interfejsu MQAI

Przedstawiona lista zawiera przykładowe programy, które demonstrują użycie interfejsu MQAI. Przykłady wykonują następujące zadania:

1. Utwórz kolejkę lokalną. "Przykładowy program w języku C do tworzenia kolejki lokalnej (amqsaicq.c)" na stronie 22
2. Wyświetlaj zdarzenia na ekranie za pomocą prostego monitora zdarzeń. "Przykładowy program w języku C do wyświetlania zdarzeń za pomocą monitora zdarzeń (amqsaiem.c)" na stronie 26
3. Służy do drukowania listy wszystkich kolejek lokalnych i ich bieżących głębokości. "Przykładowy program w języku C do uzyskiwania informacji o kolejkach i drukowaniu informacji (amqsailq.c)" na stronie 38
4. Wydrukuj listę wszystkich kanałów i ich typów. "Przykładowy program w języku C do uzyskiwania informacji o obiektach kanału (amqsaicl.c)" na stronie 33

Budowanie aplikacji MQAI

Aby zbudować aplikację przy użyciu interfejsu MQAI, należy połączyć się z tymi samymi bibliotekami, co w przypadku produktu WebSphere MQ. Informacje na temat sposobu budowania aplikacji produktu WebSphere MQ zawiera sekcja Budowanie aplikacji WebSphere MQ .

Wskazówki i wskazówki dotyczące konfigurowania produktu WebSphere MQ przy użyciu interfejsu MQAI

Interfejs MQAI używa komunikatów PCF do wysyłania komend administracyjnych do serwera komend, a nie do bezpośredniego zajmowania się tym serwerem komend. Wskazówki dotyczące konfigurowania produktu WebSphere MQ przy użyciu interfejsu MQAI można znaleźć w podręczniku "Porady i wskazówki dotyczące konfigurowania produktu IBM WebSphere MQ" na stronie 42 .

Interfejs administracyjny produktu IBM WebSphere MQ (MQAI)

Produkty IBM WebSphere MQ dla systemów Windows, AIX, Linux, HP-UXi Solaris obsługują interfejs administracyjny produktu IBM WebSphere MQ (MQAI). Interfejs MQAI to interfejs programistyczny do produktu IBM WebSphere MQ , który udostępnia alternatywę dla interfejsu MQI w celu wysyłania i odbierania PCF.

Interfejs MQAI używa *worków danych* , które umożliwiają łatwiejszą obsługę właściwości (lub parametrów) obiektów w sposób łatwiejszy niż korzystanie z systemów PCF bezpośrednio za pośrednictwem interfejsu MQAI.

Interfejs MQAI zapewnia łatwiejszy dostęp do programowego dostępu do komunikatów PCF, przekazując parametry do worka danych, tak aby dla każdej struktury wymagana była tylko jedna instrukcja. Ten dostęp usuwa potrzebę programisty do obsługi tablic i przydzielania pamięci masowej, a także zapewnia pewną izolację od szczegółów PCF.

Interfejs MQAI administruje produktem WebSphere MQ , wysyłając komunikaty PCF do serwera komend i oczekując na odpowiedź.

Interfejs MQAI jest opisany w drugiej sekcji tego podręcznika. Opis interfejsu modelu obiektu komponentu dla interfejsu MQAI można znaleźć w dokumentacji produktu Używanie języka Java .

Przykładowy program w języku C do tworzenia kolejki lokalnej (amqsaicq.c)

Przykładowy program w języku C `amqsaicq.c` tworzy kolejkę lokalną przy użyciu interfejsu MQAI.

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

/*****************************************************************************/
/*                                                                           */
/* AMQSAICQ has 2 parameters - the name of the local queue to be created     */
```

```
/*                          - the queue manager name (optional)             */
/*                                                                          */
/****************************************************************************/
/****************************************************************************/
/* Includes                                                                 */
/****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

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

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

int main(int argc, char *argv[])
{
   MQHCONN hConn;                       /* handle to WebSphere MQ connection   */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name   */
   MQLONG connReason;                   /* MQCONN reason code               */
   MQLONG compCode;                     /* completion code                  */
   MQLONG reason;                       /* reason code                      */

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

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

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


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

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

}
/****************************************************************************/
/*                                                                          */
/* Function:    CreateLocalQueue                                            */
/* Description: Create a local queue by sending a PCF command to the command  */
/*              server.                                                     */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/* Input Parameters:  Handle to the queue manager                           */
/*                    Name of the queue to be created                       */
/*                                                                          */
/* Output Parameters: None                                                  */
```

```
/*                                                                  */
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.     */
/*        The call generates the correct PCF structure.             */
/*        The default options to the call are used so that the command is sent*/
/*        to the SYSTEM.ADMIN.COMMAND.QUEUE.                        */
/*        The reply from the command server is placed on a temporary dynamic  */
/*        queue.                                                    */
/*        The reply is read from the temporary queue and formatted into the   */
/*        response bag.                                             */
/*                                                                  */
/*        The completion code from the mqExecute call is checked and if there */
/*        is a failure from the command server then the code returned by the  */
/*        command server is retrieved from the system bag that is             */
/*        embedded in the response bag to the mqExecute call.               */
/*                                                                  */
/********************************************************************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG reason;                       /* reason code                 */
   MQLONG compCode;                     /* completion code             */
   MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute      */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute     */
   MQHBAG resultBag;                    /* result bag from mqExecute       */
   MQLONG mqExecuteCC;                  /* mqExecute completion code       */
   MQLONG mqExecuteRC;                  /* mqExecute reason code           */

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


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

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

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

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

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

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

```
      CheckCallResult("MQDISC", compCode, reason);
      exit(98);
   }


   /*****************************************************************/
   /* Check the result from mqExecute call and find the error if it failed.   */
   /*****************************************************************/
   if ( compCode == MQCC_OK )
      printf("Local queue %s successfully created\n", qName);
   else
   {
      printf("Creation of local queue %s failed: Completion Code = %d
              qName, compCode, reason);
      if (reason == MQRCCF_COMMAND_FAILED)
      {
         /*****************************************************************/
         /* Get the system bag handle out of the mqExecute response bag.      */
         /* This bag contains the reason from the command server why the      */
         /* command failed.                                                   */
         /*****************************************************************/
         mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
                      &reason);
         CheckCallResult("Get the result bag handle", compCode, reason);

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

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


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

```
}
```

Przykładowy program w języku C do wyświetlania zdarzeń za pomocą monitora zdarzeń (amqsaiem.c)

Przykładowy program w języku C `amqsaiem.c` demonstruje podstawowy monitor zdarzeń przy użyciu interfejsu 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;
}
```

Przykładowy program w języku C do uzyskiwania informacji o obiektach kanału (amqsaicl.c)

Przykładowy program w języku C `amqsaicl.c` zapytuje obiekty kanału przy użyciu interfejsu 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);
}
```

Przykładowy program w języku C do uzyskiwania informacji o kolejkach i drukowaniu informacji (amqsailq.c)

Przykładowy program w języku C `amqsailq.c` pyta o bieżące zapełnienie kolejek lokalnych za pomocą interfejsu MQAI.

```


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

```
/* AMQSAILQ has 1 parameter - the queue manager name (optional)               */
/*                                                                            */
/******************************************************************************/

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

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

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

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


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

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

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

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

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

```
   CheckCallResult("Add q name", compCode, reason);

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

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

   /*****************************************************************/
   /* Send the command to find all the local queue names and queue depths.    */
   /* The mqExecute call creates the PCF structure required, sends it to      */
   /* the command server, and receives the reply from the command server into */
   /* the response bag. The attributes are contained in system bags that are  */
   /* embedded in the response bag, one set of attributes per bag.            */
   /*****************************************************************/
   mqExecute(hConn,                   /* WebSphere MQ connection handle          */
             MQCMD_INQUIRE_Q,         /* Command to be executed                  */
             MQHB_NONE,               /* No options bag                          */
             adminBag,                /* Handle to bag containing commands       */
             responseBag,             /* Handle to bag to receive the response*/
             MQHO_NONE,               /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
             MQHO_NONE,               /* Create a dynamic q for the response     */
             &compCode,               /* Completion code from the mqExecute      */
             &reason);                /* Reason code from mqExecute call         */


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

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

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

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

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

       /*************************************************************/
```

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

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

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

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

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

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

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

*********************************************************************************/
*                                                                               */
* Function: CheckCallResult                                                     */
*                                                                               */
*********************************************************************************/
*                                                                               */
* Input Parameters:  Description of call                                        */
*                    Completion code                                            */
*                    Reason code                                                */
*                                                                               */
* Output Parameters: None                                                       */
*                                                                               */
```

```
* Logic: Display the description of the call, the completion code and the     */
*        reason code if the completion code is not successful                 */
*                                                                             */
*******************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
  if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
                callText, cc, rc);
}
```

Porady i wskazówki dotyczące konfigurowania produktu IBM WebSphere MQ

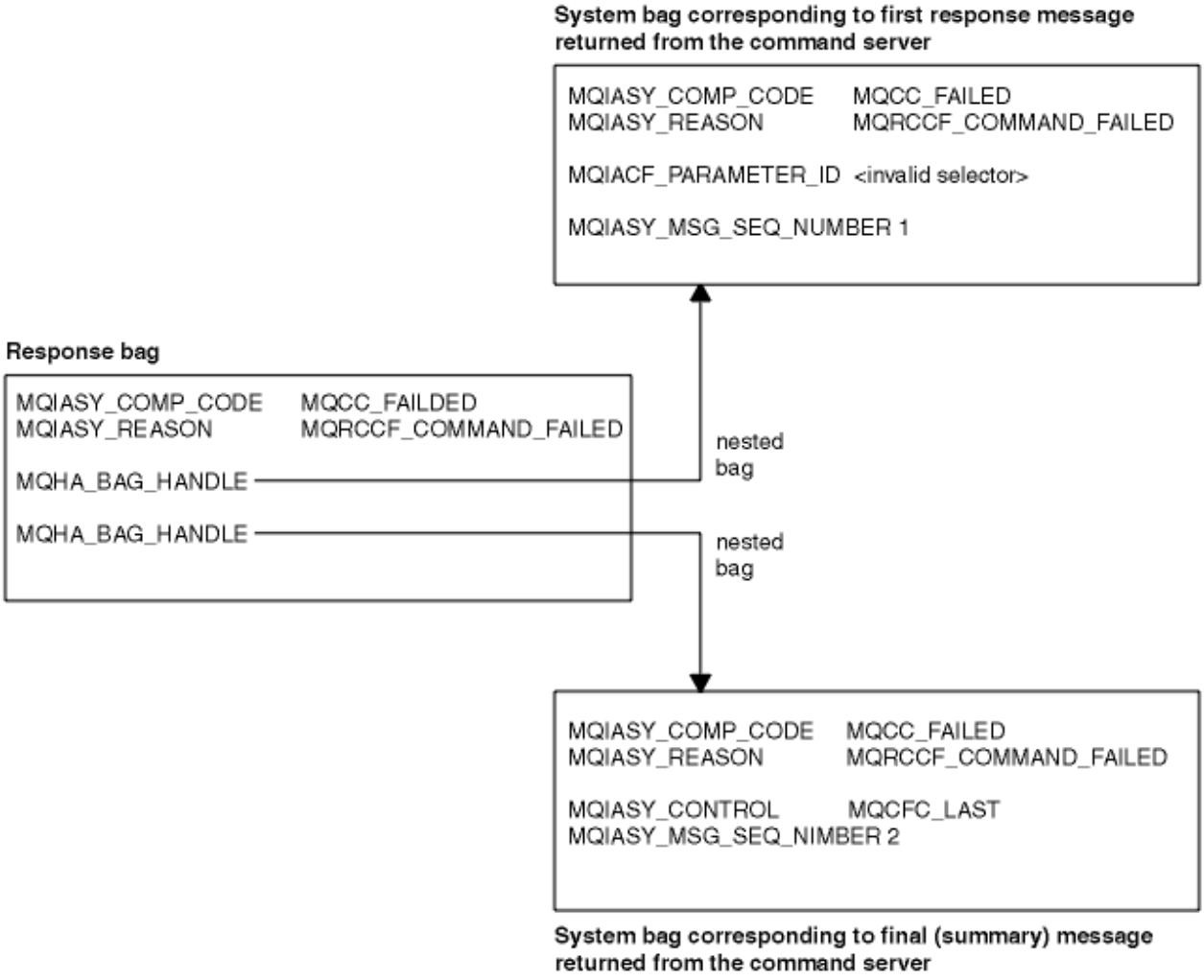
Wskazówki i wskazówki dotyczące programowania podczas korzystania z interfejsu MQAI.

Interfejs MQAI używa komunikatów PCF do wysyłania komend administracyjnych do serwera komend, a nie do bezpośredniego zajmowania się tym serwerem komend. Poniżej przedstawiono kilka wskazówek dotyczących konfigurowania produktu WebSphere MQ przy użyciu interfejsu MQAI:

- Łańcuchy znaków w produkcie WebSphere MQ są puste dopełniane do stałej długości. Za pomocą języka C łańcuchy zakończone znakiem o kodzie zero mogą być zwykle dostarczane jako parametry wejściowe do interfejsów programistycznych produktu WebSphere MQ .
- Aby wyczyścić wartość atrybutu łańcuchowego, należy ustawić go jako pojedynczy pusty łańcuch, a nie pusty łańcuch.
- Należy rozważyć z wyprzedzeniem atrybuty, które mają zostać zmienione, a także zapytania dotyczące tylko tych atrybutów.
- Niektóre atrybuty nie mogą być zmieniane, na przykład nazwa kolejki lub typ kanału. Upewnij się, że podjęto próbę zmiany tylko tych atrybutów, które mogą być modyfikowane. Zapoznaj się z listą wymaganych i opcjonalnych parametrów dla konkretnego obiektu zmiany PCF. Więcej informacji zawiera sekcja Definicje formatów komend programowalnych.
- Jeśli wywołanie MQAI nie powiedzie się, w worku odpowiedzi zostaną zwrócone szczegółowe informacje o niepowodzeniu. Dalsze szczegóły można następnie znaleźć w zagnieżdżonej torbie, do której można uzyskać dostęp za pomocą selektora MQHA_BAG_HANDLE. Na przykład, jeśli wywołanie mqExecute nie powiedzie się z kodem przyczyny MQRCCF_COMMAND_FAILED, informacje te są zwracane w worku odpowiedzi. Możliwe przyczyny tego kodu przyczyny: określony selektor nie jest poprawny dla typu komunikatu komendy, a ten szczegół informacji znajduje się w zagnieżdżonej torbie, do której można uzyskać dostęp za pomocą uchwytu torby.

  Więcej informacji na temat komendy MQExecute można znaleźć w sekcji “Wysyłanie komend administracyjnych do serwera komend przy użyciu wywołania mqExecute” na stronie 57 .

  Poniższy diagram przedstawia ten scenariusz:



Zaawansowane tematy MQAI

Informacje na temat indeksowania, konwersji danych i używania deskryptora komunikatu

- Indeksowanie

  Indeksy są używane przy zastępowaniu lub usuwaniu istniejących elementów danych z torby w celu zachowania kolejności wstawiania. Szczegółowe informacje na temat indeksowania można znaleźć w podręczniku “Indeksowanie w interfejsie MQAI” na stronie 43.

- Konwersja danych

  Łańcuchy zawarte w torbie danych MQAI mogą znajdować się w różnych kodowanych zestawach znaków i można je przekształcić za pomocą wywołania mqSetInteger. Szczegółowe informacje na temat konwersji danych można znaleźć w podręczniku “Konwersja danych w interfejsie MQAI” na stronie 44.

- Użycie deskryptora komunikatu

  Interfejs MQAI generuje deskryptor komunikatu, który jest ustawiany na wartość początkową podczas tworzenia worka danych. Szczegółowe informacje na temat używania deskryptora komunikatu można znaleźć w sekcji “Użycie deskryptora komunikatu w interfejsie MQAI” na stronie 46.

Indeksowanie w interfejsie MQAI

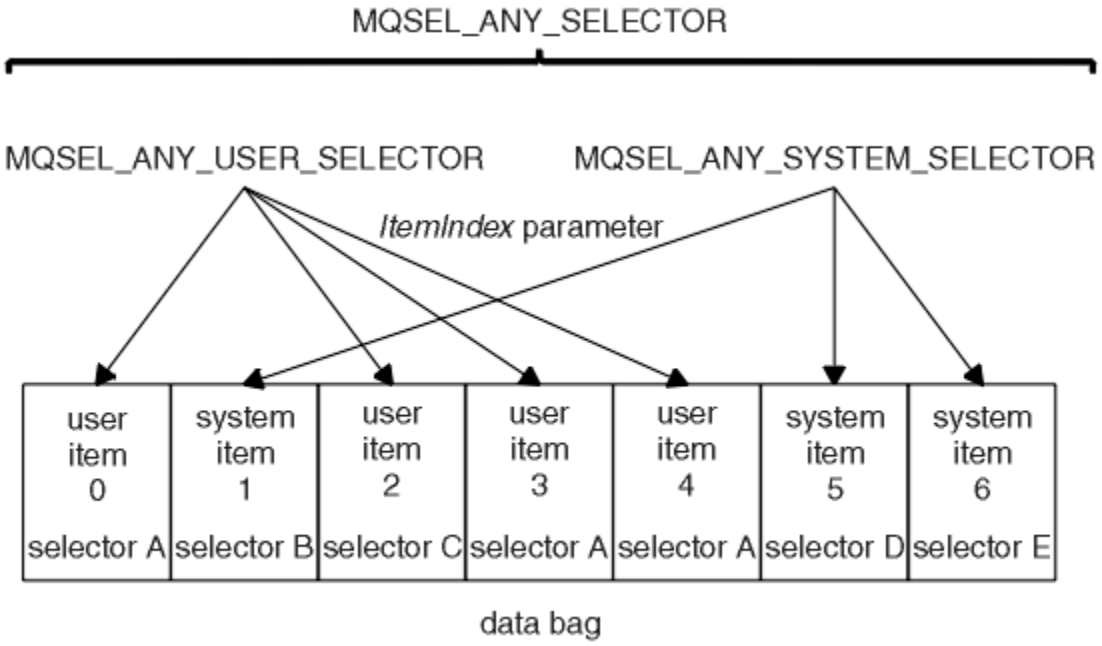
Indeksy są używane przy zastępowaniu lub usuwaniu istniejących elementów danych z torby. Istnieją trzy typy indeksowania, dzięki którym elementy danych mogą być łatwo pobrane.

Każdy selektor i wartość w obrębie elementu danych w torbie mają trzy powiązane numery indeksów:

- Indeks względem innych elementów, które mają ten sam selektor.

- Indeks w odniesieniu do kategorii selektora (użytkownika lub systemu), do którego należy element.
- Indeks względem wszystkich elementów danych w torbie (użytkownik i system).

Umożliwia to indeksowanie przez selektory użytkownika, selektory systemowe lub oba te elementy, jak to pokazano na Rysunek 1 na stronie 44.



Rysunek 1. Indeksowanie

Na rysunku Rysunek 1 na stronie 44element użytkownika 3 (selektor A) może być przywołany przez następujące pary indeksów:

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

Indeks jest oparty na zerach, podobnie jak w przypadku tablicy w języku C. Jeśli występują wystąpienia 'n' wystąpień, indeks waha się od zera do wartości'n-1', bez przerw.

Indeksy są używane przy zastępowaniu lub usuwaniu istniejących elementów danych z torby. W przypadku użycia w ten sposób kolejność wstawiania jest zachowywana, ale może mieć to wpływ na indeksy innych elementów danych. Przykłady tego można znaleźć w sekcji Zmiana informacji w torbie i Usuwanie elementów danych.

Trzy typy indeksowania umożliwiają łatwe pobieranie elementów danych. Na przykład, jeśli w worku znajdują się trzy instancje konkretnego selektora, wywołanie mqCountElementów może zliczyć liczbę instancji tego selektora, a wywołania mqInquire* mogą określać zarówno selektor, jak i indeks, aby uzyskać dostęp tylko do tych wartości. Jest to przydatne w przypadku atrybutów, które mogą mieć listę wartości, takich jak niektóre z wyjść w kanałach.

Konwersja danych w interfejsie MQAI

Łańcuchy zawarte w torbie danych MQAI mogą znajdować się w różnych kodowanych zestawach znaków. Łańcuchy te można przekształcić za pomocą wywołania mqSetInteger.

Podobnie jak w przypadku komunikatów PCF, łańcuchy zawarte w worku danych MQAI mogą znajdować się w różnych kodowanych zestawach znaków. Zwykle wszystkie łańcuchy w komunikacie PCF znajdują się w tym samym kodowanym zestawie znaków; jest to ten sam zestaw, co menedżer kolejek.

Każdy element łańcucha w worku danych zawiera dwie wartości: sam łańcuch i identyfikator CCSID. Łańcuch dodawany do tego pliku jest uzyskiwany z parametru *Buffer* w łańcuchu mqAddString lub mqSetString. Identyfikator CCSID jest uzyskiwany z elementu systemu zawierającego selektor tabeli MQIASY_CODED_CHAR_SET_ID. Jest on znany jako *identyfikator CCSID torby* i może zostać zmieniony przy użyciu wywołania mqSetInteger.

Podczas uzyskiwania informacji o wartości łańcucha zawartego w worku danych, identyfikator CCSID jest parametrem wyjściowym z wywołania.

Tabela 2 na stronie 45 przedstawia reguły stosowane podczas przekształcania worków danych w komunikaty i odwrotnie:

Tabela 2. Przetwarzanie CCSID

Wywołanie MQAI	CCSID	Dane wejściowe do wywołania	Dane wyjściowe do wywołania
mqBagToBuffer	Identyfikator CCSID torby (1)	Ignorowany	Niezmienione
mqBagToBuffer	Identyfikatory CCSID łańcuchów w torbie	Wykorzystane	Niezmienione
mqBagToBuffer	Identyfikatory CCSID łańcuchów w buforze	Nie dotyczy	Skopiowane z identyfikatorów CCSID łańcuchów w torbie
mqBufferToBag	Identyfikator CCSID torby (1)	Ignorowany	Niezmienione
mqBufferToBag	Identyfikatory CCSID łańcuchów w buforze	Wykorzystane	Niezmienione
mqBufferToBag	Identyfikatory CCSID łańcuchów w torbie	Nie dotyczy	Skopiowano z łańcuchowych identyfikatorów CCSID w buforze
TorbamqPut	CCSID MQMD	Wykorzystane	Niezmienione (2)
TorbamqPut	Identyfikator CCSID torby (1)	Ignorowany	Niezmienione
TorbamqPut	Identyfikatory CCSID łańcuchów w torbie	Wykorzystane	Niezmienione
TorbamqPut	Identyfikatory CCSID łańcuchów w wysłanym komunikacie	Nie dotyczy	Skopiowane z identyfikatorów CCSID łańcuchów w torbie
TorbamqGet	CCSID MQMD	Służy do przetwarzania danych komunikatu.	Ustawiono identyfikator CCSID zwracanych danych (3)
TorbamqGet	Identyfikator CCSID torby (1)	Ignorowany	Niezmienione
TorbamqGet	Identyfikatory CCSID łańcuchów w komunikacie	Wykorzystane	Niezmienione
TorbamqGet	Identyfikatory CCSID łańcuchów w torbie	Nie dotyczy	Skopiowano z identyfikatorów CCSID łańcucha w komunikacie

Tabela 2. Przetwarzanie CCSID (kontynuacja)

Wywołanie MQAI	CCSID	Dane wejściowe do wywołania	Dane wyjściowe do wywołania
mqExecute	Identyfikator CCSID torby żądanie-worek	Używane w przypadku żądania MQMD komunikatu żądania (4)	Niezmienione
mqExecute	Identyfikator CCSID rekordu odpowiedzi	Używane do konwersji danych komunikatu odpowiedzi (4)	Ustawiono identyfikator CCSID zwracanych danych (3)
mqExecute	Identyfikatory CCSID łańcuchów w rekordzie żądania	Używane dla komunikatu żądania	Niezmienione
mqExecute	Identyfikatory CCSID łańcuchów w rekordzie odpowiedzi	Nie dotyczy	Skopiowano z identyfikatorów CCSID łańcucha w komunikacie odpowiedzi

Uwagi:

1. Identyfikator CCSID torby jest elementem systemowym z selektorem MQIASY_CODED_CHAR_SET_ID.
2. Wartość MQCCSI_Q_MGR jest zmieniana na rzeczywisty identyfikator CCSID menedżera kolejek.
3. Jeśli zażądano konwersji danych, identyfikator CCSID zwracanych danych jest taki sam jak wartość wyjściowa. Jeśli konwersja danych nie jest żądana, zwracany identyfikator CCSID danych jest taki sam jak wartość komunikatu. Należy pamiętać, że jeśli żądana jest konwersja danych, nie jest zwracany żaden komunikat.
4. Jeśli identyfikator CCSID ma wartość MQCCSI_DEFAULT, używany jest identyfikator CCSID menedżera kolejek.

Użycie deskryptora komunikatu w interfejsie MQAI

Deskryptor komunikatu generowanego przez interfejs MQAI jest ustawiany na wartość początkową podczas tworzenia worka danych.

Typ komendy PCF jest uzyskiwany z elementu systemu z selektorem MQIASY_TYPE. Podczas tworzenia worka danych początkowa wartość tego elementu jest ustawiana w zależności od typu tworzonej przez użytkownika torby:

Tabela 3. Typ komendy PCF

Rodzaj torby	Początkowa wartość elementu MQIASY_TYPE
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_ *	MQCFT_USER,

Gdy interfejs MQAI generuje deskryptor komunikatu, wartości używane w parametrach *Format* i *MsgType* zależą od wartości elementu systemu z selektorem MQIASY_TYPE, co pokazano w sekcji Tabela 3 na stronie 46.

Tabela 4. Parametry formatu i MsgType deskryptora MQMD		
Typ komendy PCF	**Format**	**MsgType**
MQCFT_COMMAND	ADMINISTRATOR MQFMT_ADMIN	MQMT_REQUEST
RAPORT MQCFT_REPORT	ADMINISTRATOR MQFMT_ADMIN	Raport_menedżera_mQMT
MQCFT_RESPONSE	ADMINISTRATOR MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	ADMINISTRATOR MQFMT_ADMIN	MQMT_DATAGRAM
MQCFT_EVENT,	Zdarzenie MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_ *	MQFMT_PCF	MQMT_DATAGRAM

Tabela 4 na stronie 47 pokazuje, że jeśli zostanie utworzony worek administracyjny lub worek komend, *Format* deskryptora komunikatu to MQFMT_ADMIN, a *MsgType* to MQMT_REQUEST. Jest to odpowiednie dla komunikatu żądania PCF wysłanego do serwera komend, gdy oczekiwana jest odpowiedź z powrotem.

Inne parametry w deskryptorze komunikatu przyjmują wartości wyświetlane w programie Tabela 5 na stronie 47.

Tabela 5. Wartości deskryptora komunikatu	
Parametr	**Wartość**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	Patrz Tabela 4 na stronie 47
Expiry	30 sekund (uwaga “1” na stronie 48)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	zależy od identyfikatora CCSID torby (uwaga “2” na stronie 48)
Format	Patrz Tabela 4 na stronie 47
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	patrz uwaga “3” na stronie 48
ReplyToQMgr	puste

Tabela 5. Wartości deskryptora komunikatu (kontynuacja)	
Parametr	**Wartość**
Uwagi: 1. Tę wartość można przesłonić w wywołaniu **mqExecute** za pomocą parametru **OptionsBag** . Więcej informacji na ten temat zawiera sekcja **mqExecute**. 2. Patrz “Konwersja danych w interfejsie MQAI” na stronie 44. 3. Nazwa kolejki odpowiedzi podanej przez użytkownika lub kolejki tymczasowej wygenerowanej przez MQAI dla komunikatów typu **MQMT_REQUEST**. W przeciwnym razie puste.	

Torby danych

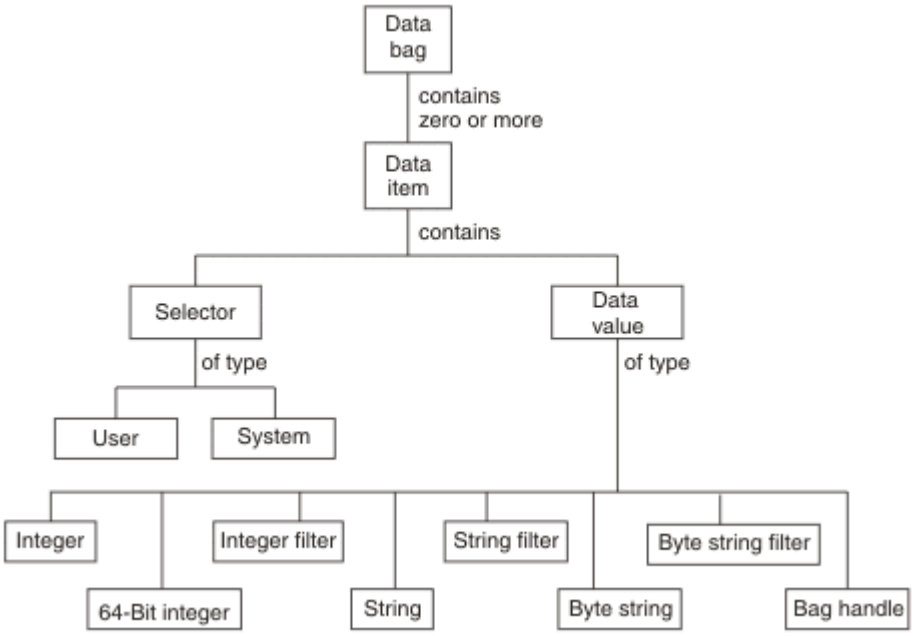
Worek danych jest środkiem do obsługi właściwości lub parametrów obiektów za pomocą interfejsu MQAI.

Torby danych

- Worek danych zawiera zero lub więcej *elementów danych*. Te elementy danych są uporządkowane w worku, ponieważ są one umieszczane w torbie. Jest to nazywane *kolejką wstawiania*. Każdy element danych zawiera *selektor* , który identyfikuje element danych i *wartość* tego elementu danych, który może być liczbą całkowitą, 64-bitową liczbą całkowitą, filtrem całkowitoliczbowym, łańcuchem, filtrem łańcuchowym, łańcuchem bajtowym, filtrem łańcucha bajtowego lub uchwytem innej torby. Elementy danych są opisane w szczegółach w sekcji “Element danych” na stronie 50

  Istnieją dwa typy selektorów: *selektory użytkownika* i *selektory systemowe*. Są one opisane w sekcji Selektory MQAI. Selektory są zwykle unikalne, ale możliwe jest, że istnieje wiele wartości dla tego samego selektora. W tym przypadku *indeks* identyfikuje konkretne wystąpienie selektora, który jest wymagany. Indeksy są opisane w sekcji “Indeksowanie w interfejsie MQAI” na stronie 43.

  Hierarchia tych pojęć jest przedstawiona na Rysunku 1.



Rysunek 2. Hierarchia pojęć związanych z MQAI

Hierarchia została wyjaśniona w poprzednim akapicie.

Typy worka danych

W zależności od zadania, które ma zostać wykonane, można wybrać typ torby danych, która ma zostać utworzona:

zbiór użytkownika
Prosta torba używana na potrzeby danych użytkownika.

zbiór administracyjny
Worek utworzony dla danych wykorzystywanych do administrowania obiektami WebSphere MQ przez wysyłanie komunikatów administracyjnych do serwera komend. Torba administracyjna automatycznie implikuje pewne opcje opisane w sekcji “Tworzenie i usuwanie worków danych” na stronie 49.

zbiór komend
W przypadku komend do administrowania obiektami WebSphere MQ tworzona jest również torba. Jednak w przeciwieństwie do worka administracyjnego, worek komend nie oznacza automatycznie niektórych opcji, mimo że opcje te są dostępne. Aby uzyskać więcej informacji na temat opcji, zobacz: “Tworzenie i usuwanie worków danych” na stronie 49.

worek grupowy
Worek używany do przechowywania zestawu zgrupowanych elementów danych. Worki grupowe nie mogą być używane do administrowania obiektami WebSphere MQ .

Dodatkowo **torba systemowa** jest tworzona przez interfejs MQAI, gdy komunikat odpowiedzi jest zwracany z serwera komend i umieszczony w worku wyjściowym użytkownika. Użytkownik nie może modyfikować torby systemowej.

W tym temacie wyszczególnia się różne sposoby używania toreb danych:

Korzystanie z worków danych

Różne sposoby korzystania z toreb danych są przedstawione na poniższej liście:

- Istnieje możliwość tworzenia i usuwania worków danych “Tworzenie i usuwanie worków danych” na stronie 49.
- Istnieje możliwość wysyłania danych między aplikacjami za pomocą worków danych “Umieszczanie i odbieranie worków danych” na stronie 50.
- Elementy danych można dodawać do toreb danych “Dodawanie elementów danych do worków” na stronie 51.
- Komendę uzyskiwania informacji można dodać w worku danych “Dodawanie komendy uzyskiwania informacji do torby” na stronie 52.
- Istnieje możliwość uzyskania informacji o danych w obrębie toreb danych “Sprawdzanie w obrębie toreb danych” na stronie 53.
- Istnieje możliwość zliczania elementów danych w obrębie worka danych “Zliczanie elementów danych” na stronie 56.
- Informacje można zmienić w worku danych “Zmiana informacji w torbie” na stronie 53.
- Użytkownik może wyczyścić worek danych “Kasowanie torby przy użyciu wywołania mqClearBag” na stronie 54.
- Można obciąć worek danych “Obcinanie torby przy użyciu wywołania funkcji mqTruncate” na stronie 55.
- Istnieje możliwość przekształcenia worków i buforów “Przekształcanie worków i buforów” na stronie 55.

Tworzenie i usuwanie worków danych

Tworzenie worków danych

Aby korzystać z interfejsu MQAI, należy najpierw utworzyć worek danych przy użyciu wywołania mqCreate. Jako dane wejściowe dla tego wywołania należy podać jedną lub więcej opcji w celu sterowania tworzeniem torby.

Parametr *Options* wywołania MQCreateBag umożliwia wybranie, czy ma zostać utworzony worek użytkownika, worek komend, worek grupowy czy też worek administracyjny.

Aby utworzyć torbę użytkownika, worek komend lub worek grupowy, można wybrać jedną lub więcej dodatkowych opcji do:

- Użyj formularza listy, gdy w worku znajdują się dwa lub więcej sąsiadujących wystąpień tego samego selektora.
- Zmień kolejność elementów danych, ponieważ są one dodawane do komunikatu PCF, aby upewnić się, że parametry są w ich poprawnej kolejności. Więcej informacji na temat elementów danych zawiera sekcja “Element danych” na stronie 50.
- Sprawdź wartości selektorów użytkownika dla elementów, które są dodawane do torby.

Worki administracyjne automatycznie oznaczają te opcje.

Worek danych jest identyfikowany przez uchwyt. Uchwyt worka jest zwracany przez komendę mqCreateBag i musi być podany we wszystkich innych wywołaniach, które korzystają z tego worka danych.

Pełny opis wywołania funkcji mqCreateBag znajduje się w sekcji mqCreateBag(Worek mqCreate).

Usuwanie worków danych

Wszystkie elementy, które są tworzone przez użytkownika, również muszą zostać usunięte przy użyciu wywołania mqDelete. Na przykład, jeśli w kodzie użytkownika jest tworzona torba, należy ją również usunąć w kodzie użytkownika.

Torby systemowe są tworzone i usuwane automatycznie przez interfejs MQAI. Więcej informacji na ten temat zawiera sekcja “Wysyłanie komend administracyjnych do serwera komend przy użyciu wywołania mqExecute” na stronie 57. Kod użytkownika nie może usunąć torby systemowej.

Pełny opis wywołania funkcji mqDeleteBag znajduje się w sekcji mqDeleteBag(Wtorka mqDelete).

Umieszczanie i odbieranie worków danych

Dane mogą być również wysyłane między aplikacjami przez wprowadzanie i pobieranie danych za pomocą wywołania mqPutBag i wywołania mqGetBag. Dzięki temu interfejs MQAI obsługuje bufor, a nie aplikację. Wywołanie mqPutBag przekształca treść podanej torby w komunikat PCF i wysyła komunikat do określonej kolejki, a wywołanie funkcji mqGetBag usuwa komunikat z określonej kolejki i przekształca je z powrotem w worek danych. Dlatego wywołanie metody mqPutBag jest odpowiednikiem wywołania mqBagToBuffer , po którym następuje wywołanie MQPUT, a element mqGetBag jest odpowiednikiem wywołania MQGET, po którym następuje wywołanie metody mqBufferToBag.

Więcej informacji na temat wysyłania i odbierania komunikatów PCF w określonej kolejce można znaleźć w sekcji “Wysyłanie i odbieranie komunikatów PCF w określonej kolejce” na stronie 12

Uwaga: Jeśli zostanie wybrana opcja użycia wywołania mqGetBag, szczegóły PCF w komunikacie muszą być poprawne; jeśli nie są, odpowiednie wyniki błędów i komunikat PCF nie jest zwracany.

Element danych

Elementy danych są używane do zapełniania worków danych podczas ich tworzenia. Tymi elementami danych mogą być elementy użytkownika lub systemowe.

Te elementy użytkownika zawierają dane użytkownika, takie jak atrybuty obiektów, które są administrowane. Elementy systemowe powinny być używane do większej kontroli nad wygenerowanymi

komunikatami: na przykład generowanie nagłówków komunikatów. Więcej informacji na temat elementów systemowych zawiera sekcja “Elementy systemowe” na stronie 51.

Typy elementów danych

Po utworzeniu worka danych można je zapełnić elementami całkowitymi lub łańcuchami znakowymi. Można zapytać o wszystkie trzy typy pozycji.

Element danych może być liczbą całkowitą lub pozycją łańcucha znaków. Poniżej przedstawiono typy elementów danych dostępne w obrębie interfejsu MQAI:

- Liczba całkowita
- 64-bitowa liczba całkowita
- Filtr liczb całkowitych
- Znakowy-łańcuch
- Filtr łańcuchowy
- Łańcuch bajtów
- Filtr łańcucha bajtowego
- Uchwyt worka

Korzystanie z elementów danych

Są to następujące sposoby korzystania z elementów danych:

- “Zliczanie elementów danych” na stronie 56.
- “Usuwanie elementów danych” na stronie 56.
- “Dodawanie elementów danych do worków” na stronie 51.
- “Filtrowanie i odpytywanie elementów danych” na stronie 52.

Elementy systemowe

Elementy systemowe mogą być używane dla:

- Generowanie nagłówków PCF. Elementy systemu mogą sterować identyfikatorem komendy PCF, opcjami sterowania, numerem kolejnym komunikatu i typem komendy.
- Konwersja danych. Elementy systemu obsługują identyfikator zestawu znaków dla pozycji łańcucha znaków w torbie.

Podobnie jak wszystkie elementy danych, elementy systemowe składają się z selektora i wartości. Więcej informacji na temat tych selektorów i ich wartości zawiera sekcja Selektory MQAI.

Elementy systemowe są unikalne. Jeden lub więcej elementów systemu może być identyfikowany przez selektor systemu. Istnieje tylko jedno wystąpienie każdego selektora systemu.

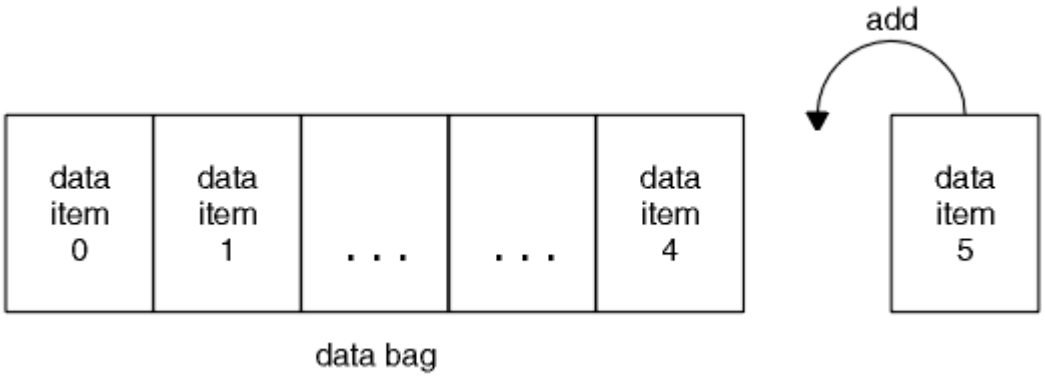
Większość elementów systemu może być modyfikowana (patrz “Zmiana informacji w torbie” na stronie 53), ale użytkownik nie może zmienić opcji tworzenia worka. Nie można usuwać elementów systemowych. (Patrz “Usuwanie elementów danych” na stronie 56).

Dodawanie elementów danych do worków

Gdy tworzona jest torba danych, można ją zapełnić elementami danych. Tymi elementami danych mogą być elementy użytkownika lub systemowe. Więcej informacji na temat elementów danych zawiera sekcja “Element danych” na stronie 50.

Zmaterializowana tabela zapytania (MQAI) umożliwia dodawanie pozycji całkowitych, 64-bitowych elementów całkowitych, elementów filtru liczb całkowitych, elementów łańcucha znakowego, filtru łańcuchów, elementów łańcucha bajtowego i elementów filtru łańcucha bajtów do worków, co jest wyświetlane w sekcji Rysunek 3 na stronie 52. Elementy są identyfikowane przez selektor. Zwykle jeden selektor identyfikuje tylko jeden element, ale nie zawsze jest to przypadek. Jeśli element danych

o określonym selektorze znajduje się już w torbie, dodatkowa instancja tego selektora jest dodawana do końca torby.



Rysunek 3. Dodawanie elementów danych

Dodaj elementy danych do torby przy użyciu wywołań mqAdd*:

- Aby dodać elementy całkowite, należy użyć wywołania mqAddInteger w sposób opisany w sekcji mqAddInteger .
- Aby dodać 64-bitowe elementy całkowite, należy użyć wywołania mqAddInteger64 zgodnie z opisem w sekcji mqAddInteger64 .
- Aby dodać elementy filtru liczby całkowitej, należy użyć wywołania mqAddIntegerFilter zgodnie z opisem w sekcji mqAddIntegerFilter .
- Aby dodać elementy znakowe, należy użyć wywołania mqAddString zgodnie z opisem w sekcji mqAddString .
- Aby dodać elementy filtru łańcucha, należy użyć wywołania mqAddStringFilter zgodnie z opisem w sekcji mqAddStringFilter .
- Aby dodać elementy łańcucha bajtowego, należy użyć wywołania mqAddByteString zgodnie z opisem w sekcji mqAddByteString .
- Aby dodać elementy filtru łańcuchów bajtów, należy użyć wywołania filtru mqAddByteStringzgodnie z opisem w sekcji mqAddByteStringFilter .

Aby uzyskać więcej informacji na temat dodawania elementów danych do torby, patrz “Elementy systemowe” na stronie 51.

Dodawanie komendy uzyskiwania informacji do torby

Wywołanie zapytania mqAddsłuży do dodawania komendy uzyskiwania informacji do torby. Połączenie jest specjalnie przeznaczone do celów administracyjnych, dlatego może być używane tylko z workami administracyjnymi. Umożliwia ona określenie selektorów atrybutów, dla których użytkownik chce uzyskać informacje z produktu WebSphere MQ.

Pełny opis wywołania uzyskiwania informacji o produkcie mqAddmożna znaleźć w sekcji ZapytaniemqAdd.

Filtrowanie i odpytywanie elementów danych

Podczas korzystania z interfejsu MQAI do uzyskiwania informacji o atrybutach obiektów WebSphere MQ można sterować danymi zwracaną do programu na dwa sposoby.

- Można ***filtrować*** danych zwracanych za pomocą wywołań mqAddInteger i mqAddString. To podejście umożliwia określenie pary *Selector* i *ItemValue* , na przykład:

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

  Ten przykład określa, że typ kolejki (*Selector*) musi być lokalny (*ItemValue*), a ta specyfikacja musi być zgodna z atrybutami obiektu (w tym przypadku kolejką), o której użytkownik jest pytany.

Inne atrybuty, które mogą być filtrowane, odpowiadają komendom PCF Inquire *, które można znaleźć w programie “Wprowadzenie do formatów komend programowalnych” na stronie 9. Na przykład, aby dowiedzieć się więcej o atrybutach kanału, należy zapoznać się z informacjami na temat komendy Inquire Channel w tej dokumentacji produktu. Parametry "Wymagane parametry" i "Opcjonalne" w komendzie Inquire Channel identyfikują selektory, których można użyć do filtrowania.

- Istnieje możliwość ***zapytania*** poszczególnych atrybutów obiektu za pomocą wywołania zapytania mqAdd. Określa selektor, w którym użytkownik jest zainteresowany. Jeśli selektor nie zostanie określony, zwrócone zostaną wszystkie atrybuty obiektu.

Poniżej znajduje się przykład filtrowania i wysyłania zapytań do atrybutów kolejki:

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

Więcej przykładów filtrowania i odpylania elementów danych zawiera sekcja “Przykłady użycia interfejsu MQAI” na stronie 21.

Sprawdzanie w obrębie toreb danych

Informacje na temat:

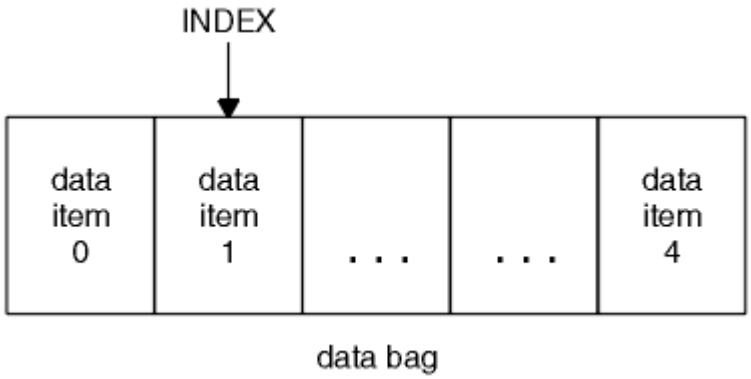
- Wartość elementu całkowitoliczbowego przy użyciu wywołania mqInquireInteger. Patrz mqInquireInteger.
- Wartość 64-bitowego elementu całkowitoliczbowego przy użyciu wywołania mqInquireInteger64 . Patrz mqInquireInteger64.
- Wartość elementu filtru całkowitoliczbowego za pomocą wywołania mqInquireIntegerFilter . Patrz mqInquireIntegerFilter.
- Wartość elementu łańcucha znakowego przy użyciu wywołania łańcucha mqInquire. Patrz mqInquireString.
- Wartość elementu filtru łańcucha przy użyciu wywołania mqInquireStringFilter . Patrz mqInquireStringFilter.
- Wartość elementu łańcucha bajtowego przy użyciu wywołania mqInquireByteString . Patrz mqInquireByteString.
- Wartość elementu filtru łańcucha bajtowego przy użyciu wywołania filtru mqInquireByteString. Patrz mqInquireByteStringFilter.
- Wartość uchwytu torby przy użyciu wywołania funkcji mqInquire. Patrz mqInquireBag.

Można również zapytać o typ (liczba całkowita, 64-bitowa liczba całkowita, filtr liczb całkowitych, łańcuch znaków, filtr łańcuchowy, łańcuch bajtowy, filtr łańcucha bajtowego lub uchwyt worka) konkretnego elementu przy użyciu wywołania mqInquireItemInfo . Patrz mqInquireItemInfo.

Zmiana informacji w torbie

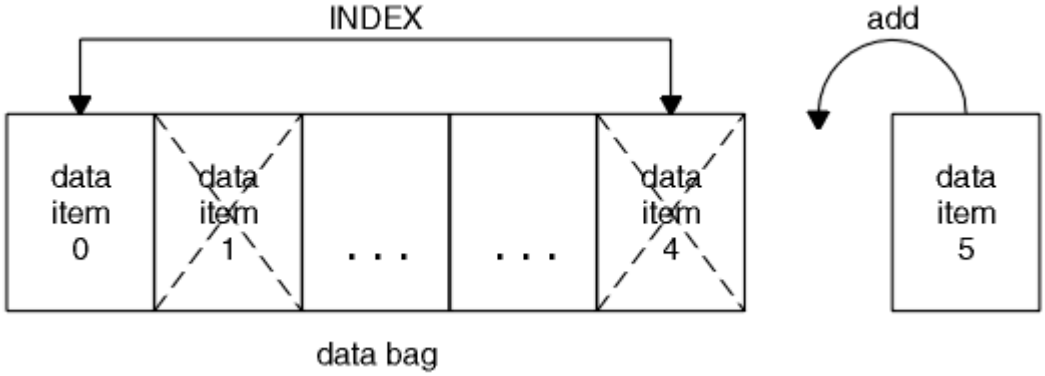
Interfejs MQAI umożliwia zmianę informacji w obrębie torby przy użyciu wywołań mqSet*. Można wykonać następujące czynności:

1. Modyfikowanie elementów danych w torbie. Indeks umożliwia zastąpienie pojedynczej instancji parametru identyfikacją wystąpienia elementu, który ma zostać zmodyfikowany (patrz Rysunek 4 na stronie 54).



Rysunek 4. Modyfikowanie pojedynczego elementu danych

2. Usuń wszystkie istniejące wystąpienia określonego selektora i dodaj nowe wystąpienie do końca torby. (Patrz Rysunek 5 na stronie 54.) Specjalna wartość indeksu umożliwia zastępowanie ***wszystkich*** instancji parametru.



Rysunek 5. Modyfikowanie wszystkich elementów danych

Uwaga: Indeks zachowuje kolejność wstawiania w worku, ale może wpływać na indeksy innych elementów danych.

Wywołanie mqSetLiczba całkowita umożliwia modyfikowanie elementów liczby całkowitej w obrębie torby. Wywołanie mqSetInteger64 pozwala na zmodyfikowanie 64-bitowych elementów całkowitych. Wywołanie mqSetIntegerFilter umożliwia modyfikowanie elementów filtru liczby całkowitej. Wywołanie łańcucha mqSetumożliwia modyfikowanie elementów łańcucha znaków. Wywołanie metody mqSetStringFilter umożliwia modyfikowanie elementów filtru łańcuchów. Wywołanie metody mqSetByteString umożliwia modyfikowanie elementów łańcucha bajtowego. Wywołanie filtru mqSetByteStringumożliwia modyfikowanie elementów filtru łańcuchów bajtów. Można również użyć tych wywołań, aby usunąć wszystkie istniejące wystąpienia określonego selektora i dodać nowe wystąpienie na końcu tej torby. Element danych może być elementem użytkownika lub elementem systemowym.

Pełny opis tych wywołań znajduje się w:

- mqSetLiczba całkowita
- mqSetInteger64
- mqSetIntegerFilter
- mqSetŁańcuch
- mqSetStringFilter
- mqSetByteString
- Filtr mqSetByteString

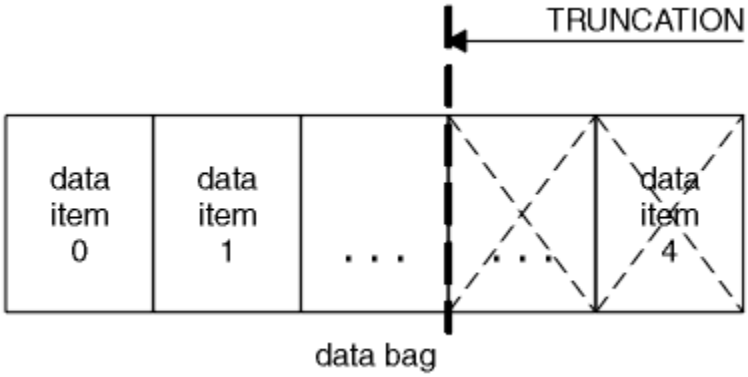
Kasowanie torby przy użyciu wywołania mqClearBag

Wywołanie funkcji mqClearBag usuwa wszystkie elementy użytkownika z worka użytkownika i resetuje elementy systemowe do ich wartości początkowych. Worki systemowe zawarte w worku są również usuwane.

Pełny opis wywołania funkcji mqClearBag znajduje się w sekcji mqClearBag(Torba mqClear).

Obcinanie torby przy użyciu wywołania funkcji mqTruncate

Wywołanie funkcji mqTruncateumożliwia zmniejszenie liczby elementów użytkownika w torbie użytkownika przez usunięcie elementów z końca torby, zaczynając od ostatnio dodanego elementu. Na przykład można użyć tego samego nagłówka w celu wygenerowania więcej niż jednego komunikatu.

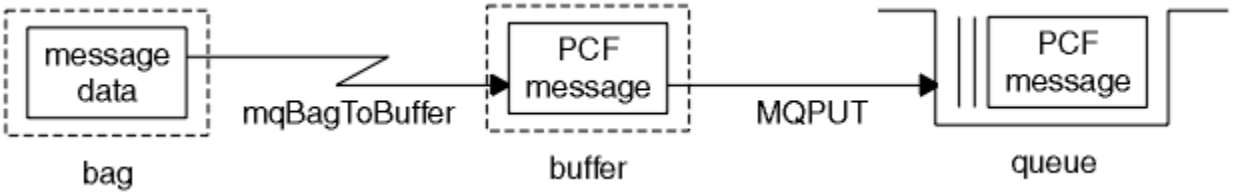


Rysunek 6. Obcinanie torby

Pełny opis wywołania funkcji mqTruncateBag znajduje się w sekcji mqTruncateBag.

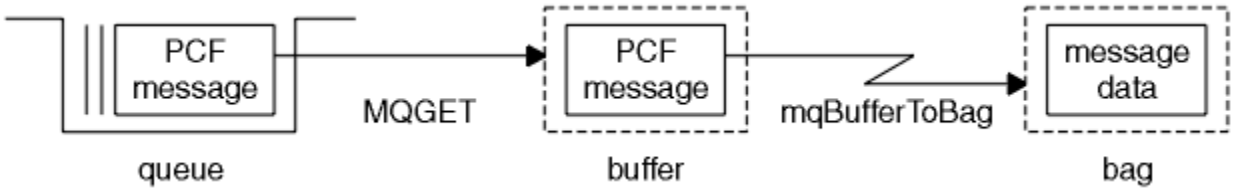
Przekształcanie worków i buforów

Aby wysłać dane między aplikacjami, po pierwsze dane komunikatu są umieszczane w torbie. Następnie dane znajdujące się w torbie są przekształcane w komunikat PCF za pomocą wywołania mqBagToBuffer . Komunikat PCF jest wysyłany do wymaganej kolejki przy użyciu wywołania MQPUT. Pokazano to na rysunku Rysunek 7 na stronie 55. Pełny opis wywołania mqBagToBuffer można znaleźć w sekcji mqBagToBuffer.



Rysunek 7. Przekształcanie toreb w komunikaty PCF

Aby odbierać dane, komunikat jest odbierany do buforu przy użyciu wywołania MQGET. Dane w buforze są następnie przekształcane w torbę za pomocą wywołania mqBufferToBag , udostępniając bufor, który zawiera poprawny komunikat PCF. Pokazano to na rysunku Rysunek 8 na stronie 55. Pełny opis wywołania mqBufferToBag można znaleźć w sekcji mqBufferToBag.



Rysunek 8. Przekształcanie komunikatów PCF w formularz torby

Zliczanie elementów danych

Wywołanie mqCountPozycje zlicza liczbę elementów użytkownika, elementów systemowych lub obu tych elementów, które są przechowywane w torbie danych, i zwraca ten numer. Na przykład, `mqCountItems(Bag, 7, ...)`, zwraca liczbę elementów w torbie z selektorem 7. Może on zliczać pozycje według indywidualnego selektora, selektory użytkownika, selektory systemowe lub wszystkie selektory.

Uwaga: To wywołanie zlicza liczbę elementów danych, a nie liczbę unikalnych selektorów w torbie. Selektor może występować wiele razy, więc w worku może być mniej unikalnych selektorów niż elementy danych.

Pełny opis wywołania mqCountItems zawiera sekcja mqCountItems.

Usuwanie elementów danych

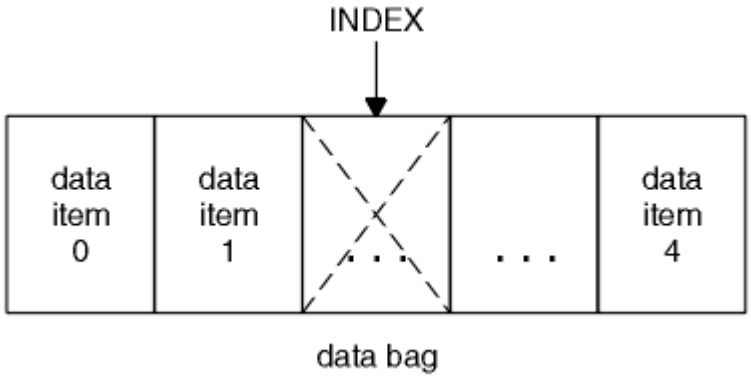
Elementy z worków można usuwać na wiele sposobów. Można wykonać następujące czynności:

- Usuń jeden lub więcej elementów użytkownika z torby. Więcej informacji można znaleźć w sekcji “Usuwanie elementów danych z torby przy użyciu wywołania elementu mqDelete” na stronie 56.
- Usuń ***wszystkie*** elementy użytkownika z torby, to znaczy *Wyczyść* torbę. Szczegółowe informacje na ten temat zawiera sekcja “Kasowanie torby przy użyciu wywołania mqClearBag” na stronie 54.
- Usuń elementy użytkownika z końca torby, to znaczy *obetnij* worek. Więcej informacji można znaleźć w sekcji “Obcinanie torby przy użyciu wywołania funkcji mqTruncate” na stronie 55.

Usuwanie elementów danych z torby przy użyciu wywołania elementu mqDelete

Wywołanie elementu mqDeletepowoduje usunięcie jednego lub większej liczby elementów użytkownika z torby. Indeks jest używany do usuwania:

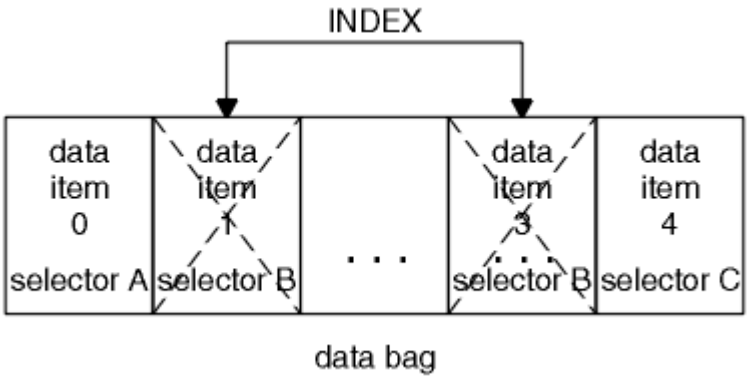
1. Pojedyncze wystąpienie określonego selektora. (Patrz Rysunek 9 na stronie 56.)



Rysunek 9. Usuwanie pojedynczego elementu danych

lub wersji

2. Wszystkie wystąpienia określonego selektora. (Patrz Rysunek 10 na stronie 57.)



Rysunek 10. Usuwanie wszystkich elementów danych

Uwaga: Indeks zachowuje kolejność wstawiania w worku, ale może wpływać na indeksy innych elementów danych. Na przykład wywołanie elementu mqDeletenie zachowuje wartości indeksu dla elementów danych, które są zgodne z usuniętym elementem, ponieważ indeksy są reorganizowane w celu wypełnienia luki, która pozostaje z usuniętego elementu.

Pełny opis wywołania elementu mqDeletemożna znaleźć w sekcji mqDeleteItem(Element mqDelete).

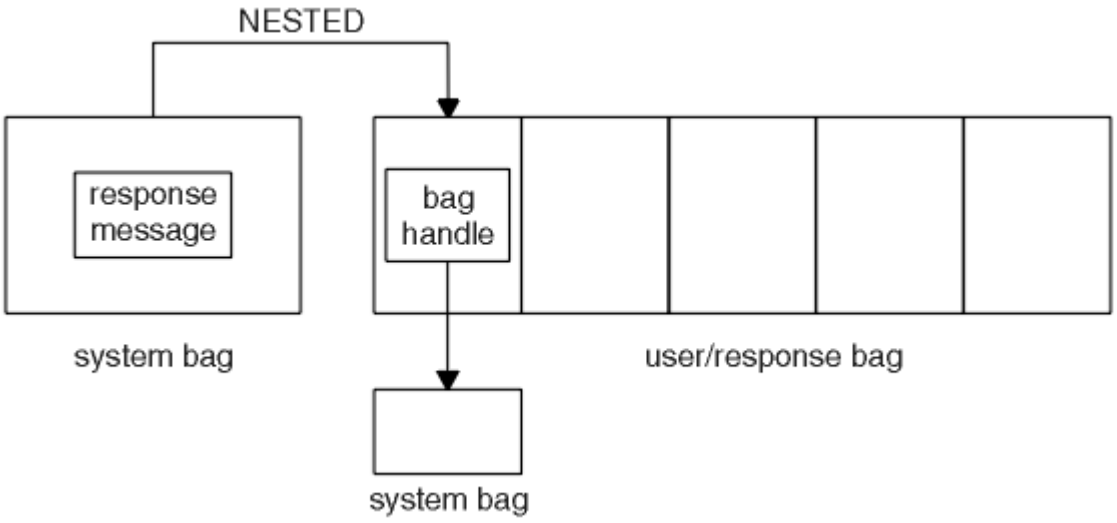
Wysyłanie komend administracyjnych do serwera komend przy użyciu wywołania mqExecute

Po utworzeniu i zapełnieniu worka danych do serwera komend menedżera kolejek za pomocą wywołania mqExecute może zostać wysłany komunikat komendy administracyjnej. Ta opcja obsługuje wymianę z serwerem komend i zwraca odpowiedzi w torbie.

Po utworzeniu i zapełnieniu worka danych można wysłać komunikat komendy administracyjnej do serwera komend menedżera kolejek. Najprostszym sposobem wykonania tego działania jest użycie wywołania mqExecute . Wywołanie funkcji mqExecute wysyła komunikat komendy administracyjnej jako komunikat nietrwały i oczekuje na wszystkie odpowiedzi. Odpowiedzi są zwracane w worku odpowiedzi. Mogą one zawierać informacje o atrybutach związanych z kilkoma obiektami WebSphere MQ lub serią komunikatów o błędach PCF, na przykład. W związku z tym worek odpowiedzi może zawierać tylko kod powrotu lub może zawierać *worki zagnieżdżone*.

Komunikaty odpowiedzi są umieszczane w workach systemowych, które są tworzone przez system. Na przykład w przypadku zapytań dotyczących nazw obiektów tworzona jest torba systemowa, w której przechowywane są te nazwy obiektów, a torebka jest wstawiana do worka użytkownika. Uchwyty do tych worków są następnie wstawiane do worka odpowiedzi, a do zagnieżdżonej torby można uzyskać dostęp za pomocą selektora MQHA_BAG_HANDLE. Torba systemowa pozostaje w pamięci masowej, jeśli nie została usunięta, dopóki nie zostanie usunięta torba odpowiedzi.

Pojęcie *zagnieżdżania* jest przedstawione w sekcji Rysunek 11 na stronie 58.



Rysunek 11. Zagnieżdżenie

Jako dane wejściowe dla wywołania mqExecute należy podać:

- Uchwyt połączenia MQI.
- Komenda do wykonania. Powinna to być jedna z wartości MQCMD_ *.

  Uwaga: Jeśli ta wartość nie jest rozpoznawana przez interfejs MQAI, wartość jest nadal akceptowana. Jeśli jednak do wstawiania wartości do worka użyto wywołania mqAddInquiry, ten parametr musi być komendą INQUIRE rozpoznawaną przez interfejs MQAI. Oznacza to, że parametr powinien mieć postać MQCMD_INQUIRE_ *.
- Opcjonalnie uchwyt worka zawierający opcje sterujące przetwarzaniem wywołania. W tym miejscu można również określić maksymalny czas (w milisekundach), przez który program MQAI powinien czekać na każdy komunikat odpowiedzi.
- Uchwyt worka administracyjnego, który zawiera szczegóły komendy administracyjnej, która ma zostać wydana.
- Uchwyt worka odpowiedzi, który odbiera komunikaty odpowiedzi.

Następujące elementy są opcjonalne:

- Uchwyt obiektu kolejki, w której ma zostać umieszczona komenda administracyjna.

  Jeśli nie zostanie podany żaden uchwyt obiektu, komenda administracyjna zostanie umieszczona w systemie SYSTEM.ADMIN.COMMAND.QUEUE należącej do aktualnie połączonego menedżera kolejek. Jest to opcja domyślna.
- Uchwyt obiektu kolejki, w którym mają być umieszczane komunikaty odpowiedzi.

  Komunikaty odpowiedzi można umieścić w kolejce dynamicznej, która jest tworzona automatycznie przez interfejs MQAI. Utworzona kolejka istnieje tylko przez czas trwania wywołania i jest usuwana przez interfejs MQAI przy wyjściu z wywołania mqExecute .

Przykłady użycia wywołania komendy mqExecute można znaleźć w sekcji Przykładowy kod .

Administrowanie przy użyciu IBM WebSphere MQ Explorer

IBM WebSphere MQ Explorer umożliwia lokalne lub zdalne administrowanie siecią za pomocą komputera z uruchomionym systemem Windowslub systemem Linux (platformyx86 i x86-64).

IBM WebSphere MQ dla produktów Windowsi IBM WebSphere MQ for Linux (platformyx86 i x86-64) udostępniają interfejs administracyjny o nazwie IBM WebSphere MQ Explorer , który umożliwia wykonywanie zadań administracyjnych jako alternatywa dla używania komend sterujących lub MQSC. Sekcja Porównywanie zestawów komend zawiera informacje o tym, co można zrobić za pomocą programu IBM WebSphere MQ Explorer.

IBM WebSphere MQ Explorer umożliwia lokalne lub zdalne administrowanie siecią na podstawie komputera z uruchomionym systemem Windowslub Linux (platformyx86-64), wskazując IBM WebSphere MQ Explorer w menedżerach kolejek i klastrach, które są zainteresowane. Platformy i poziomy partycji IBM WebSphere MQ , które można administrować za pomocą IBM WebSphere MQ Explorer , są opisane w sekcji “Zdalne menedżery kolejek” na stronie 60.

Aby skonfigurować zdalne menedżery kolejek produktu IBM WebSphere MQ , tak aby produkt IBM WebSphere MQ Explorer mógł administrować nimi, należy zapoznać się z “Wstępnie wymagane oprogramowanie i definicje” na stronie 61.

Umożliwia on wykonywanie zadań, zwykle związanych z konfigurowaniem i dostrajaniem środowiska pracy dla produktu IBM WebSphere MQ, lokalnie lub zdalnie w domenie systemowej Windows lub Linux (platformyx86 i x86-64).

W systemie Linuxuruchomienie IBM WebSphere MQ Explorer może się nie powieść, jeśli użytkownik ma więcej niż jedną instalację Eclipse . W takim przypadku należy uruchomić IBM WebSphere MQ Explorer przy użyciu innego identyfikatora użytkownika, który będzie używany dla innej instalacji Eclipse .

W systemie Linux, aby pomyślnie uruchomić serwer IBM WebSphere MQ Explorer , należy mieć możliwość zapisu pliku w katalogu osobistym, a katalog osobisty musi istnieć.

Co można zrobić za pomocą IBM WebSphere MQ Explorer

Poniżej znajduje się lista zadań, które można wykonać za pomocą konsoli IBM WebSphere MQ Explorer.

Za pomocą programu IBM WebSphere MQ Explorer można:

- Tworzenie i usuwanie menedżera kolejek (tylko na komputerze lokalnym).
- Uruchamianie i zatrzymywanie menedżera kolejek (tylko na komputerze lokalnym).
- Zdefiniuj, wyświetl i zmień definicje obiektów produktu WebSphere MQ , takich jak kolejki i kanały.
- Przeglądaj komunikaty w kolejce.
- Uruchamianie i zatrzymywanie kanału.
- Wyświetlanie informacji o statusie kanału, obiektu nasłuchiwania, kolejki lub obiektów usług.
- Wyświetlanie menedżerów kolejek w klastrze.
- Sprawdź, które aplikacje, użytkownicy lub kanały mają otwartą określoną kolejkę.
- Utwórz nowy klaster menedżera kolejek przy użyciu kreatora *Utwórz nowy klaster* .
- Dodaj menedżera kolejek do klastra za pomocą kreatora *Dodawanie menedżera kolejek do klastra* .
- Zarządzanie obiektem informacji uwierzytelniającej, używanym przez zabezpieczenia kanału SSL (Secure Sockets Layer).
- Tworzenie i usuwanie inicjatorów kanałów, monitorów wyzwalaczy i programów nasłuchujących.
- Uruchamianie lub zatrzymywanie serwerów komend, inicjatorów kanałów, monitorów wyzwalaczy i programów nasłuchujących.
- Ustaw konkretne usługi, które mają być uruchamiane automatycznie podczas uruchamiania menedżera kolejek.
- Zmodyfikuj właściwości menedżerów kolejek.
- Zmień lokalny domyślny menedżer kolejek.
- Za pomocą interfejsu GUI programu ikeyman można zarządzać certyfikatami SSL (Secure Sockets Layer), powiązać certyfikaty z menedżerami kolejek oraz skonfigurować i skonfigurować składnice certyfikatów (tylko na komputerze lokalnym).
- Utwórz obiekty JMS z obiektów WebSphere MQ i obiektów WebSphere MQ z obiektów JMS.
- Utwórz fabrykę połączeń JMS dla dowolnego z obecnie obsługiwanych typów.
- Zmodyfikuj parametry dla dowolnej usługi, takie jak numer portu TCP dla obiektu nasłuchiwania lub nazwę kolejki inicjatora kanału.
- Uruchom lub zatrzymaj śledzenie usługi.

Zadania administracyjne można wykonywać przy użyciu szeregu *Widoki treści* i *okien dialogowych właściwości*.

Widok zawartości
Widok treści to panel, który może wyświetlać następujące elementy:

- Atrybuty i opcje administracyjne odnoszące się do produktu WebSphere MQ .
- Atrybuty i opcje administracyjne odnoszące się do jednego lub większej liczby powiązanych obiektów.
- Atrybuty i opcje administracyjne dla klastra.

Okna dialogowe właściwości
Okno dialogowe właściwości to panel wyświetlający atrybuty odnoszące się do obiektu z serii pól, z których niektóre można edytować.

Nawigowanie za pomocą programu WebSphere MQ Explorer odbywa się za pomocą *widokuNavigator*. Navigator umożliwia wybranie widoku treści, który jest wymagany.

Zdalne menedżery kolejek

Istnieją dwa wyjątki od obsługiwanych menedżerów kolejek, z którymi można nawiązać połączenie.

Z poziomu systemu Windows lub Linux (platformyx86 i x86-64) program WebSphere MQ Explorer może łączyć się ze wszystkimi obsługiwanymi menedżerami kolejek z następującymi wyjątkami:

- WebSphere MQ dla menedżerów kolejek w systemie z/OS wcześniejszych niż wersja 6.0.
- Obecnie obsługiwane menedżery kolejek MQSeries V2 .

Eksplorator produktu IBM WebSphere MQ obsługuje różnice w możliwościach między różnymi poziomami komend i platformami. Jeśli jednak napotka atrybut, który nie rozpoznaje atrybutu, atrybut nie będzie widoczny.

Jeśli planowane jest zdalne administrowanie menedżerem kolejek w wersji V6.0 lub nowszej w systemie Windows za pomocą programu IBM WebSphere MQ Explorer na komputerze z produktem WebSphere MQ V5.3 , należy zainstalować pakiet poprawek Fix Pack 9 (CSD9) lub nowszy na komputerze z produktem WebSphere MQ for Windows V5.3 .

Jeśli użytkownik zamierza zdalnie administrować menedżerem kolejek V5.3 w systemie iSeries za pomocą programu WebSphere MQ Explorer na komputerze z produktem WebSphere MQ V6.0 lub nowszym, Należy zainstalować pakiet poprawek Fix Pack 11 (CSD11) lub nowszy na komputerze z produktem WebSphere MQ for iSeries V5.3 . Ten pakiet poprawek koryguje problemy związane z połączeniem między programem WebSphere MQ Explorer i menedżerem kolejek iSeries .

Rozstrzyganie, czy ma być używany IBM WebSphere MQ Explorer

Podczas określania, czy podczas instalacji ma być używany produkt IBM WebSphere MQ Explorer , należy wziąć pod uwagę informacje wymienione w tym temacie.

Należy zdawać sobie sprawę z następujących kwestii:

nazwy obiektów
Jeśli podczas pracy z obiektami za pomocą komend MQSC używane są małe nazwy dla menedżerów kolejek i innych obiektów za pomocą programu IBM WebSphere MQ Explorer, nazwy obiektów należy ująć w pojedynczy cudzysłów lub WebSphere MQ nie rozpoznaje tych obiektów.

Duże menedżery kolejek
Eksplorator IBM WebSphere MQ działa najlepiej z małymi menedżerami kolejek. Jeśli w jednym menedżerze kolejek istnieje duża liczba obiektów, mogą wystąpić opóźnienia, gdy program WebSphere MQ Explorer wyodrębnia wymagane informacje w celu wyświetlenia w widoku.

Klastry
Klastry WebSphere MQ mogą potencjalnie zawierać setki lub tysiące menedżerów kolejek. Program WebSphere MQ Explorer przedstawia menedżery kolejek w klastrze przy użyciu struktury drzewa. Fizyczna wielkość klastra nie wpływa na szybkość działania programu IBM WebSphere MQ Explorer,

ponieważ program IBM WebSphere MQ Explorer nie łączy się z menedżerami kolejek w klastrze, dopóki nie zostaną wybrane.

Konfigurowanie produktu IBM WebSphere MQ Explorer

W tej sekcji opisano kroki, które należy wykonać, aby skonfigurować produkt IBM WebSphere MQ Explorer.

- "Wstępnie wymagane oprogramowanie i definicje" na stronie 61
- "Zabezpieczenia" na stronie 61
- "Wyświetlanie i ukrywanie menedżerów kolejek i klastrów" na stronie 65
- "Przypisania do klastrów" na stronie 66
- "Konwersja danych" na stronie 66

Wstępnie wymagane oprogramowanie i definicje

Przed podjęciem próby użycia IBM WebSphere MQ Explorernależy upewnić się, że spełnione są następujące wymagania.

Program IBM WebSphere MQ Explorer może łączyć się ze zdalnymi menedżerami kolejek tylko przy użyciu protokołu komunikacyjnego TCP/IP.

Sprawdź, czy:

1. Serwer komend jest uruchomiony dla każdego zdalnie administrowanego menedżera kolejek.
2. Odpowiedni obiekt nasłuchiwania TCP/IP musi być uruchomiony w każdym zdalnym menedżerze kolejek. Ten obiekt może być programem nasłuchującym IBM WebSphere MQ lub, w systemach UNIX and Linux , demonem inetd.
3. Kanał połączenia z serwerem, domyślnie nazwany SYSTEM.ADMIN.SVRCONN, istnieje we wszystkich menedżerach kolejek zdalnych.

   Kanał można utworzyć za pomocą następującej komendy MQSC:

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

   Ta komenda tworzy podstawową definicję kanału. Jeśli potrzebna jest bardziej zaawansowana definicja (na przykład w celu skonfigurowania zabezpieczeń), potrzebne są dodatkowe parametry. Więcej informacji na ten temat zawiera sekcja DEFINE CHANNEL .
4. Kolejka systemowa SYSTEM.MQEXPLORER.REPLY.MODELmusi istnieć.

Zabezpieczenia

Jeśli produkt WebSphere MQ jest używany w środowisku, w którym ważne jest, aby sterować dostępem użytkowników do określonych obiektów, może być konieczne rozważenie aspektów zabezpieczeń przy użyciu programu IBM WebSphere MQ Explorer.

Autoryzacja do używania IBM WebSphere MQ Explorer

Każdy użytkownik może korzystać z IBM WebSphere MQ Explorer, ale niektóre uprawnienia są wymagane do nawiązywania połączeń, uzyskiwania dostępu do menedżerów kolejek i zarządzania nimi.

Aby wykonać lokalne zadania administracyjne przy użyciu programu WebSphere MQ Explorer, użytkownik musi mieć uprawnienia niezbędne do wykonywania zadań administracyjnych. Jeśli użytkownik jest członkiem grupy mqm , użytkownik ma uprawnienia do wykonywania wszystkich lokalnych zadań administracyjnych.

Aby nawiązać połączenie ze zdalnym menedżerem kolejek i wykonać zdalne zadania administracyjne za pomocą programu WebSphere MQ Explorer, użytkownik wykonujący program WebSphere MQ Explorer musi mieć następujące uprawnienia:

- Uprawnienie CONNECT do obiektu docelowego menedżera kolejek

- Uprawnienie INQUIRE do obiektu docelowego menedżera kolejek
- Wyświetlaj uprawnienia do docelowego obiektu menedżera kolejek
- INQUIRE do kolejki, SYSTEM.MQEXPLORER.REPLY.MODEL
- Wyświetlaj uprawnienia do kolejki SYSTEM.MQEXPLORER.REPLY.MODEL
- INPUT (get) uprawnienie do kolejki, SYSTEM.MQEXPLORER.REPLY.MODEL
- Uprawnienie OUTPUT (put) do kolejki SYSTEM.ADMIN.COMMAND.QUEUE
- INQUIRE w kolejce, SYSTEM.ADMIN.COMMAND.QUEUE
- Uprawnienie do wykonania wybranego działania

Uwaga: Uprawnienie INPUT odnosi się do danych wejściowych dla użytkownika z kolejki (operacja get). Uprawnienia OUTPUT odnoszą się do danych wyjściowych użytkownika do kolejki (operacja put).

Aby połączyć się ze zdalnym menedżerem kolejek w produkcie WebSphere MQ for z/OS i wykonać zdalne zadania administracyjne za pomocą Eksploratora IBM WebSphere MQ , należy podać następujące informacje:

- Profil RACF dla kolejki systemowej SYSTEM.MQEXPLORER.REPLY.MODEL
- Profil RACF dla kolejek AMQ.MQEXPLORER.*

Ponadto użytkownik uruchamiający program WebSphere MQ Explorer musi mieć następujące uprawnienia:

- Uprawnienie RACF UPDATE do kolejki systemowej SYSTEM.MQEXPLORER.REPLY.MODEL
- Uprawnienie RACF UPDATE do kolejek, AMQ.MQEXPLORER.*
- Uprawnienie CONNECT do obiektu docelowego menedżera kolejek
- Uprawnienie do wykonania wybranego działania
- Uprawnienie READ do wszystkich profili hlq.DISPLAY.object w klasie MQCMDS

Informacje na temat nadawania uprawnień do obiektów WebSphere MQ zawiera sekcja Udzielanie dostępu do obiektu WebSphere MQ w systemach UNIX lub Linux oraz w systemie Windows.

Jeśli użytkownik podejmie próbę wykonania operacji, której nie ma do wykonania, docelowy menedżer kolejek wywołuje procedury dotyczące niepowodzenia autoryzacji, a operacja nie powiedzie się.

Domyślnym filtrem w programie WebSphere MQ Explorer jest wyświetlanie wszystkich obiektów produktu WebSphere MQ . Jeśli istnieją jakiekolwiek obiekty WebSphere MQ , do których użytkownik nie ma uprawnienia do wyświetlania, generowane są niepowodzenia autoryzacji. Jeśli rejestrowane są zdarzenia uprawnień, należy ograniczyć zakres obiektów wyświetlanych do tych obiektów, do których użytkownik ma uprawnienie DISPLAY.

Zabezpieczenia łączenia ze zdalnymi menedżerami kolejek

Należy zabezpieczyć kanał między programem IBM WebSphere MQ Explorer i każdym zdalnym menedżerem kolejek.

Program IBM WebSphere MQ Explorer łączy się ze zdalnymi menedżerami kolejek jako aplikacją kliencką MQI. Oznacza to, że każdy zdalny menedżer kolejek musi mieć definicję kanału połączenia z serwerem i odpowiedni program nasłuchujący TCP/IP. Jeśli kanał połączenia z serwerem nie jest bezpieczny, istnieje możliwość nawiązania przez złośliwą aplikację połączenia z tym samym kanałem połączenia z serwerem i uzyskania dostępu do obiektów menedżera kolejek z nieograniczonym uprawnieniami. W celu zabezpieczenia kanału połączenia z serwerem należy określić niepustą wartość atrybutu MCAUSER kanału, użyć rekordów uwierzytelniania kanału lub użyć wyjścia zabezpieczeń.

Domyślna wartość atrybutu MCAUSER to identyfikator użytkownika lokalnego. Jeśli jako atrybut MCAUSER kanału połączenia z serwerem zostanie podana niepusta nazwa użytkownika, wszystkie programy łączące się z menedżerem kolejek przy użyciu tego kanału będą uruchamiane z tożsamością nazwanego użytkownika i mają ten sam poziom uprawnień. Nie jest to możliwe, jeśli używane są rekordy uwierzytelniania kanału.

Korzystanie z wyjścia zabezpieczeń przy użyciu programu WebSphere MQ Explorer

Za pomocą programu WebSphere MQ Explorer można określić domyślne wyjścia zabezpieczeń specyficzne dla programu obsługi wyjścia zabezpieczeń i menedżera kolejek.

Użytkownik może zdefiniować domyślne wyjście zabezpieczeń, które może być używane dla wszystkich nowych połączeń klienckich z programu WebSphere MQ Explorer. To domyślne wyjście może zostać przesłonięte w momencie nawiązania połączenia. Istnieje również możliwość zdefiniowania wyjścia zabezpieczeń dla pojedynczego menedżera kolejek lub zestawu menedżerów kolejek, które to wyjście jest wykonywane po nawiązaniu połączenia. Procedury zewnętrzne są określane przy użyciu programu WebSphere MQ Explorer. Więcej informacji na ten temat zawiera Centrum pomocy produktu WebSphere MQ .

Korzystanie z IBM WebSphere MQ Explorer w celu nawiązania połączenia ze zdalnym menedżerem kolejek za pomocą kanałów MQI z obsługą SSL

IBM WebSphere MQ Explorer łączy się ze zdalnymi menedżerami kolejek przy użyciu kanału MQI. Aby zabezpieczyć kanał MQI za pomocą zabezpieczeń SSL, należy utworzyć kanał przy użyciu tabeli definicji kanału klienta.

Informacje na temat tworzenia kanału MQI przy użyciu tabeli definicji kanału klienta zawiera sekcja Przegląd klientów MQI produktu IBM WebSphere MQ.

Gdy kanał jest ustanawiany przy użyciu tabeli definicji kanału klienta, można użyć programu IBM WebSphere MQ Explorer w celu nawiązania połączenia ze zdalnym menedżerem kolejek przy użyciu kanału MQI obsługującego protokół SSL, zgodnie z opisem w sekcji “Zadania w systemie, w którym znajduje się zdalny menedżer kolejek” na stronie 63 i “Zadania w systemie, w którym znajduje się IBM WebSphere MQ Explorer” na stronie 63.

Zadania w systemie, w którym znajduje się zdalny menedżer kolejek

W systemie, w którym znajduje się zdalny menedżer kolejek, wykonaj następujące czynności:

1. Zdefiniuj parę kanałów połączenia z serwerem i połączenia z klientem, a następnie określ odpowiednią wartość zmiennej *SSLCIPH* w połączeniu z serwerem na obu kanałach. Więcej informacji na temat zmiennej *SSLCIPH* zawiera sekcja Zabezpieczanie kanałów za pomocą protokołu SSL .
2. Wyślij tabelę definicji kanału `AMQCLCHL.TAB` , która znajduje się w katalogu `@ipcc` menedżera kolejek, do systemu, w którym znajduje się IBM WebSphere MQ Explorer.
3. Uruchom program nasłuchujący TCP/IP na wyznaczonym porcie.
4. Umieść zarówno certyfikat CA, jak i osobiste certyfikaty SSL w katalogu SSL menedżera kolejek:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` dla systemów UNIX and Linux
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` dla systemów Windows

   Gdzie +QMNAME+ to znacznik reprezentujący nazwę menedżera kolejek.
5. Utwórz plik bazy danych kluczy typu CMS o nazwie `key.kdb` . Hasło należy ukrywać w pliku, zaznaczając opcję w interfejsie GUI programu iKeyman lub używając opcji `-stash` z komendami **`runmqckm`** .
6. Dodaj certyfikaty ośrodka CA do bazy danych kluczy utworzonej w poprzednim kroku.
7. Zaimportuj certyfikat osobisty dla menedżera kolejek do bazy danych kluczy.

Więcej szczegółowych informacji na temat pracy z protokołem Secure Sockets Layer w systemach Windows zawiera sekcja Praca z protokołem SSL lub TLS w systemach UNIX, Linux i Windows .

Zadania w systemie, w którym znajduje się IBM WebSphere MQ Explorer

W systemie, w którym znajduje się IBM WebSphere MQ Explorer, wykonaj następujące czynności:

1. Utwórz plik bazy danych kluczy typu JKS o nazwie `key.jks`. Ustaw hasło dla tego pliku bazy danych kluczy.

Program IBM WebSphere MQ Explorer korzysta z plików kluczy Java (JKS) w celu zabezpieczenia SSL, dlatego plik kluczy tworzony na potrzeby konfigurowania protokołu SSL dla IBM WebSphere MQ Explorer musi być zgodny z tym plikiem.

2. Dodaj certyfikaty ośrodka CA do bazy danych kluczy utworzonej w poprzednim kroku.
3. Zaimportuj certyfikat osobisty dla menedżera kolejek do bazy danych kluczy.
4. W systemach Windows i Linux uruchom program MQ Explorer, korzystając z menu systemowego, pliku wykonywalnego `MQExplorer` lub komendy **strmqcfg** .
5. Na pasku narzędzi programu IBM WebSphere MQ Explorer kliknij opcję **Okno-> Preferencje**, a następnie rozwiń pozycję **Eksplorator WebSphere MQ** i kliknij przycisk **Składnice certyfikatów klienta SSL**. Wprowadź nazwę i hasło dla pliku JKS utworzonego w kroku 1 programu "Zadania w systemie, w którym znajduje się IBM WebSphere MQ Explorer" na stronie 63, zarówno w magazynie zaufanych certyfikatów, jak i w osobistej bazie certyfikatów, a następnie kliknij przycisk **OK**.
6. Zamknij okno **Preferencje** , a następnie kliknij prawym przyciskiem myszy opcję **Menedżery kolejek**. Kliknij opcję **Pokaż/ukryj menedżery kolejek**, a następnie kliknij przycisk **Dodaj** na ekranie **Pokaż/ukryj menedżery kolejek** .
7. Wpisz nazwę menedżera kolejek i wybierz opcję **Połącz bezpośrednio** . Kliknij przycisk Dalej.
8. Wybierz opcję **Użyj tabeli definicji kanału klienta (CCDT)** i określ położenie pliku tabeli kanału, który został przesłany ze zdalnego menedżera kolejek w kroku 2 w sekcji "Zadania w systemie, w którym znajduje się zdalny menedżer kolejek" na stronie 63 w systemie, w którym znajduje się zdalny menedżer kolejek.
9. Kliknij przycisk **Zakończ**. Teraz można uzyskać dostęp do menedżera kolejek zdalnych z poziomu IBM WebSphere MQ Explorer.

Nawiązanie połączenia przez inny menedżer kolejek

Program IBM WebSphere MQ Explorer umożliwia nawiązanie połączenia z menedżerem kolejek za pośrednictwem pośredniego menedżera kolejek, do którego program IBM WebSphere MQ Explorer jest już połączony.

W takim przypadku program IBM WebSphere MQ Explorer umieszcza komunikaty komend PCF w pośrednim menedżerze kolejek, podając następujące informacje:

- Parametr *ObjectQMgrNazwa* w deskryptorze obiektu (MQOD) jako nazwa docelowego menedżera kolejek. Więcej informacji na temat rozstrzygania nazw kolejek można znaleźć w sekcji Uchwała nazwy .
- Parametr *UserIdentifier* w deskryptorze komunikatu (MQMD) jako lokalny userId.

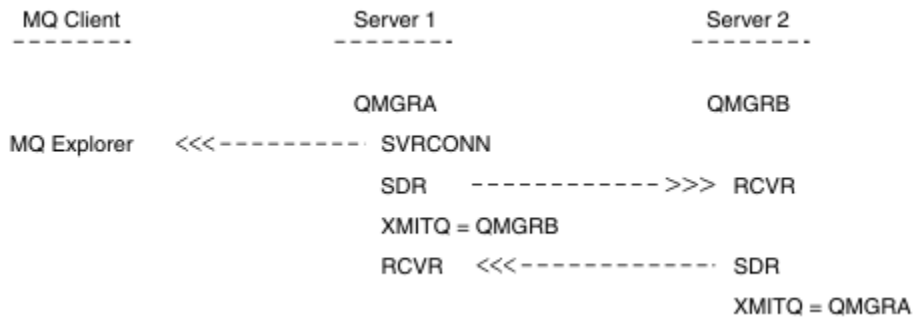
Jeśli połączenie jest następnie używane do łączenia się z docelowym menedżerem kolejek za pośrednictwem pośredniego menedżera kolejek, to userId jest ponownie wysyłany w parametrze *UserIdentifier* deskryptora komunikatu (MQMD). Aby program nasłuchujący MCA w docelowym menedżerze kolejek akceptować ten komunikat, musi być ustawiony atrybut MCAUSER lub parametr userId musi już istnieć z uprawnieniem do umieszczania.

Serwer komend w docelowym menedżerze kolejek umieszcza komunikaty w kolejce transmisji, określając parametr userId w parametrze *UserIdentifier* w deskryptorze komunikatu (MQMD). Aby ta operacja została pomyślnie wykonana, użytkownik userId musi już istnieć w docelowym menedżerze kolejek z uprawnieniem do umieszczania.

W poniższym przykładzie przedstawiono sposób łączenia menedżera kolejek za pośrednictwem pośredniego menedżera kolejek z programem WebSphere MQ Explorer.

Nawiąże zdalne połączenie administracyjne z menedżerem kolejek. Sprawdź, czy:

- Menedżer kolejek na serwerze jest aktywny i ma zdefiniowany kanał połączenia z serwerem (SVRCONN).
- Program nasłuchujący jest aktywny.
- Serwer komend jest aktywny.
- SYSTEM.MQ EXPLORER.REPLY.MODEL została utworzona i użytkownik ma wystarczające uprawnienia.
- Programy nasłuchujące menedżera kolejek, serwery komend i kanały nadawcze są uruchomione.



W tym przykładzie:

- Program IBM WebSphere MQ Explorer jest połączony z menedżerem kolejek QMGRA (działającym na serwerze Server1) przy użyciu połączenia klienckiego.
- Menedżer kolejek QMGRB na serwerze Server2 może być teraz połączony z programem IBM WebSphere MQ Explorer przy użyciu pośredniego menedżera kolejek (QMGRA)
- Podczas nawiązywania połączenia z produktem QMGRB za pomocą programu WebSphere MQ Explorer wybierz opcję QMGRA jako pośredni menedżer kolejek.

W tej sytuacji nie ma bezpośredniego połączenia z programem QMGRB z programu IBM WebSphere MQ Explorer; połączenie z programem QMGRB jest dostępne za pośrednictwem produktu QMGRA.

Menedżer kolejek QMGRB na serwerze Server2 jest połączony z serwerem QMGRA na serwerze Server1 za pomocą kanałów nadawczych. Kanał między QMGRA i QMGRB musi być skonfigurowany w taki sposób, aby zdalne administrowanie było możliwe. Patrz sekcja “Przygotowywanie kanałów i kolejek transmisji do zdalnego administrowania” na stronie 111.

Wyświetlanie i ukrywanie menedżerów kolejek i klastrów

Program IBM WebSphere MQ Explorer może jednocześnie wyświetlać więcej niż jeden menedżer kolejek. Na panelu Pokaż/ukryj menedżera kolejek (wybierany z menu dla węzła drzewa Menedżery kolejek) można wybrać, czy wyświetlane są informacje o innej (zdalnej) maszynie. Lokalne menedżery kolejek są wykrywane automatycznie.

Aby wyświetlić zdalny menedżer kolejek:

1. Kliknij prawym przyciskiem myszy węzeł drzewa `Queue Managers` , a następnie wybierz opcję *Pokaż/ukryj menedżery kolejek ...*.
2. Kliknij przycisk **Add** (Dodaj). Zostanie wyświetlony panel Pokaż/ukryj menedżery kolejek.
3. Wprowadź nazwę zdalnego menedżera kolejek oraz nazwę hosta lub adres IP w udostępnionych polach.

   Nazwa hosta lub adres IP są używane do nawiązania połączenia klienckiego ze zdalnym menedżerem kolejek przy użyciu domyślnego kanału połączenia z serwerem SYSTEM.ADMIN.SVRCONNlub kanału połączenia z serwerem zdefiniowanym przez użytkownika.
4. Kliknij opcję **Zakończ**.

Na panelu Pokaż/ukryj menedżery kolejek również wyświetlana jest lista wszystkich widocznych menedżerów kolejek. Panel ten służy do ukrywania menedżerów kolejek w widoku nawigacyjnym.

Jeśli w programie IBM WebSphere MQ Explorer wyświetlany jest menedżer kolejek, który jest elementem klastra, klaster zostanie wykryty i wyświetlony automatycznie.

Aby wyeksportować listę menedżerów kolejek zdalnych z tego panelu, wykonaj następujące czynności:

1. Zamknij panel Pokaż/ukryj menedżery kolejek.
2. Kliknij prawym przyciskiem myszy górny węzeł drzewa **IBM WebSphere MQ** na panelu nawigacyjnym programu WebSphere MQ Explorer, a następnie wybierz opcję **Eksportuj ustawienia programu MQ Explorer** .

3. Kliknij opcję **MQ Explorer > Ustawienia programu MQ Explorer** .
4. Wybierz opcję **Informacje o połączeniu > Menedżery kolejek zdalnych**.
5. Wybierz plik, w którym mają zostać zapisane wyeksportowane ustawienia.
6. Na koniec należy kliknąć przycisk **Zakończ** , aby wyeksportować informacje o połączeniu zdalnego menedżera kolejek do określonego pliku.

Aby zaimportować listę menedżerów kolejek zdalnych:

1. Kliknij prawym przyciskiem myszy górny węzeł drzewa **IBM WebSphere MQ** na panelu nawigacyjnym programu WebSphere MQ Explorer, a następnie wybierz opcję **Importuj ustawienia programu MQ Explorer** .
2. Kliknij opcję **MQ Explorer > Ustawienia programu MQ Explorer** .
3. Kliknij przycisk **Przeglądaj**i przejdź do ścieżki do pliku zawierającego informacje o połączeniu z menedżerem kolejek zdalnych.
4. Kliknij przycisk **Otwórz**. Jeśli plik zawiera listę menedżerów kolejek zdalnych, to pole **Informacje o połączeniu > Menedżery kolejek zdalnych** jest zaznaczone.
5. Na koniec należy kliknąć przycisk **Zakończ** , aby zaimportować informacje o połączeniu zdalnego menedżera kolejek do programu WebSphere MQ Explorer.

Przypisania do klastrów

Program IBM WebSphere MQ Explorer wymaga informacji o menedżerach kolejek, które są elementami klastra.

Jeśli menedżer kolejek jest elementem klastra, to węzeł drzewa klastra zostanie zapełniony automatycznie.

Jeśli menedżery kolejek stają się elementami klastrów podczas działania programu IBM WebSphere MQ Explorer, należy zachować program IBM WebSphere MQ Explorer z aktualnymi danymi administracyjnymi o klastrach, tak aby mógł on skutecznie komunikować się z nimi i wyświetlać poprawne informacje o klastrze w razie zażądania. Aby to zrobić, program WebSphere MQ Explorer musi mieć następujące informacje:

- Nazwa menedżera kolejek repozytorium
- Nazwa połączenia menedżera kolejek repozytorium, jeśli znajduje się on w zdalnym menedżerze kolejek

W przypadku tych informacji program WebSphere MQ Explorer może:

- Za pomocą menedżera kolejek repozytorium można uzyskać listę menedżerów kolejek w klastrze.
- Administruj menedżerami kolejek, które są elementami klastra, i znajdują się na obsługiwanych platformach i poziomach komend.

Podawanie nie jest możliwe, jeśli:

- Wybrane repozytorium stanie się niedostępne. Program WebSphere MQ Explorer nie przełącza się automatycznie na alternatywne repozytorium.
- Nie można skontaktować się z wybranym repozytorium za pośrednictwem protokołu TCP/IP.
- Wybrane repozytorium działa w menedżerze kolejek, który działa na poziomie platformy i komendy, a nie jest obsługiwany przez program WebSphere MQ Explorer.

Elementy klastra, które mogą być administrowane, mogą być lokalne lub mogą być zdalne, jeśli można się z nim skontaktować za pomocą protokołu TCP/IP. Program IBM WebSphere MQ Explorer łączy się z menedżerami kolejek lokalnych, które są elementami klastra bezpośrednio, bez połączenia z klientem.

Konwersja danych

Program IBM WebSphere MQ Explorer pracuje z identyfikatorem CCSID 1208 (UTF-8). Dzięki temu program IBM WebSphere MQ Explorer może poprawnie wyświetlać dane ze zdalnych menedżerów kolejek. Niezależnie od tego, czy połączenie z menedżerem kolejek jest nawiązane bezpośrednio, czy za

pomocą pośredniego menedżera kolejek, program IBM WebSphere MQ Explorer wymaga, aby wszystkie przychodzące komunikaty były przekształcane w identyfikator CCSID 1208 (UTF-8).

Jeśli użytkownik spróbuje nawiązać połączenie między programem IBM WebSphere MQ Explorer i menedżerem kolejek z identyfikatorem CCSID, którego program IBM WebSphere MQ Explorer nie rozpoznaje, generowany jest komunikat o błędzie.

Obsługiwane konwersje są opisane w sekcji Konwersja stron kodowych.

Zabezpieczenia w systemie Windows

Kreator przygotowania produktu WebSphere MQ tworzy specjalne konto użytkownika, dzięki czemu usługa Windows może być współużytkowana przez procesy, które muszą z niego korzystać.

Usługa Windows jest współużytkowana przez procesy klienta dla instalacji produktu IBM WebSphere MQ . Dla każdej instalacji tworzona jest jedna usługa. Każda usługa ma nazwę `MQ_`*`InstallationName`*, a jej nazwa wyświetlana jest `IBM WebSphere MQ(`*`InstallationName`*`)`. Przed Version 7.1, z tylko jedną instalacją na serwerze, pojedyncza usługa Windows została nazwana `MQSeriesServices` o nazwie wyświetlanej `IBM MQSeries`.

Ze względu na to, że każda usługa musi być współużytkowana przez sesje logowania nieinteraktywne i interaktywne, należy uruchomić każdą z nich na specjalnym koncie użytkownika. Można użyć jednego specjalnego konta użytkownika dla wszystkich usług lub utworzyć inne specjalne konta użytkowników. Każde specjalne konto użytkownika musi mieć uprawnienia użytkownika do "logowania się jako usługa", aby uzyskać więcej informacji na temat “Uprawnienia użytkownika wymagane w przypadku usługi IBM WebSphere MQ Windows” na stronie 68. Jeśli ID użytkownika nie ma uprawnień do uruchomienia usługi, usługa nie zostanie uruchomiona i w dzienniku zdarzeń systemowych Windows zostanie zwrócony błąd. Zwykle kreator przygotowania produktu IBM WebSphere MQ zostanie uruchomiony i poprawnie skonfigurowany zostanie identyfikator użytkownika. Jeśli jednak ID użytkownika został skonfigurowany ręcznie, możliwe jest rozwiązanie problemu, który będzie musiał zostać rozwiązany.

Po zainstalowaniu produktu IBM WebSphere MQ i uruchomieniu kreatora przygotowania produktu IBM WebSphere MQ po raz pierwszy zostanie utworzone lokalne konto użytkownika dla usługi o nazwie MUSR_MQADMIN z wymaganymi ustawieniami i uprawnieniami, w tym "Logowanie w trybie usługi".

W kolejnych instalacjach kreator przygotowania IBM WebSphere MQ tworzy konto użytkownika o nazwie MUSR_MQADMINx, gdzie x jest kolejnym dostępnym numerem reprezentującym identyfikator użytkownika, który nie istnieje. Hasło dla MUSR_MQADMINx jest generowane losowo podczas tworzenia konta i jest używane do konfigurowania środowiska logowania dla usługi. Wygenerowane hasło nie traci ważności.

To konto IBM WebSphere MQ nie ma wpływu na strategie kont, które są skonfigurowane w systemie w celu wymagania, aby hasła kont zostały zmienione po pewnym czasie.

Hasło nie jest znane poza tym jednorazowym przetwarzaniem i jest przechowywane przez system operacyjny Windows w zabezpieczonej części rejestru.

Korzystanie z katalogu Active Directory (tylko systemWindows)

W niektórych konfiguracjach sieciowych, w których konta użytkowników są definiowane w kontrolerach domeny korzystających z opcji Active Directory, lokalne konto użytkownika IBM WebSphere MQ może nie mieć uprawnień wymaganych do wysyłania zapytań o członkostwo w grupie innych kont użytkowników domeny. Kreator przygotowania produktu IBM WebSphere MQ określa, czy jest to przypadek, przeprowadzając testy i pytając użytkownika o konfigurację sieci.

Jeśli lokalne konto użytkownika IBM WebSphere MQ nie ma wymaganych uprawnień, kreator przygotowania produktu IBM WebSphere MQ wyświetli użytkownikowi informacje o szczegółach konta użytkownika domeny z określonymi prawami użytkownika. W przypadku uprawnień użytkownika, które są wymagane przez konto użytkownika domeny, należy zapoznać się z “Uprawnienia użytkownika wymagane w przypadku usługi IBM WebSphere MQ Windows” na stronie 68. Po wprowadzeniu przez użytkownika poprawnych szczegółów konta dla konta użytkownika domeny w kreatorze przygotowania produktu IBM WebSphere MQ konfiguruje ona usługę IBM WebSphere MQ Windows , która ma być uruchamiana

w ramach nowego konta. Szczegóły konta są przechowywane w zabezpieczonej części rejestru i nie mogą być odczytane przez użytkowników.

Gdy usługa jest uruchomiona, usługa produktu IBM WebSphere MQ Windows jest uruchamiana i pozostaje uruchomiona tak długo, jak długo usługa jest uruchomiona. Administrator produktu IBM WebSphere MQ , który loguje się na serwerze po uruchomieniu usługi Windows , może za pomocą konsoli IBM WebSphere MQ Explorer administrować menedżerami kolejek na serwerze. Łączy to IBM WebSphere MQ Explorer z istniejącym procesem usługi Windows . Te dwa działania wymagają różnych poziomów uprawnień, zanim będą mogły pracować:

- Proces uruchamiania wymaga uprawnień uruchamiania.
- Administrator produktu IBM WebSphere MQ wymaga uprawnień dostępu.

Uprawnienia użytkownika wymagane w przypadku usługi IBM WebSphere MQ Windows

Tabela w tym temacie zawiera listę praw użytkownika wymaganych do lokalnego i domenowego konta użytkownika, w ramach którego działa usługa Windows dla instalacji produktu IBM WebSphere MQ .

Zaloguj się jako zadanie wsadowe	Aktywuje usługę IBM WebSphere MQ Windows , która ma być uruchamiana w ramach tego konta użytkownika.
Zaloguj się jako usługa	Umożliwia użytkownikom ustawienie usługi IBM WebSphere MQ Windows w celu zalogowania się przy użyciu skonfigurowanego konta.
Zamknij system	Umożliwia usłudze IBM WebSphere MQ Windows zrestartowanie serwera, jeśli jest on skonfigurowany w taki sposób, gdy odtwarzanie usługi nie powiedzie się.
zwiększanie limitu miejsca na dysku	Wymagany w przypadku wywołania systemu operacyjnego `CreateProcessAsUser` .
działanie jako część systemu operacyjnego	Wymagany w przypadku wywołania systemu operacyjnego `LogonUser` .
Pomijanie sprawdzania przeglądania	Wymagany w przypadku wywołania systemu operacyjnego `LogonUser` .
zamiana znacznika poziomu procesu	Wymagany w przypadku wywołania systemu operacyjnego `LogonUser` .

Uwaga: W środowiskach działających na ASP i w aplikacjach IIS mogą być wymagane prawa do programów debugowania.

Konto użytkownika domeny musi mieć ustawione uprawnienia użytkownika Windows jako efektywne prawa użytkownika wymienione w aplikacji Zasady zabezpieczeń lokalnych. Jeśli nie, ustaw je za pomocą aplikacji Zasady zabezpieczeń lokalnych lokalnie na serwerze lub za pomocą domeny aplikacji Zabezpieczenia domeny.

Zmiana nazwy użytkownika powiązanej z usługą IBM WebSphere MQ

Może być konieczna zmiana nazwy użytkownika powiązanej z usługą IBM WebSphere MQ z MUSR_MQADMIN na inną. (Na przykład, może być konieczne, aby menedżer kolejek był powiązany z produktem DB2, który nie akceptuje nazw użytkowników dłuższych niż 8 znaków).

Procedura

1. Utwórz nowe konto użytkownika (na przykład **NEW_NAME**)
2. Użyj kreatora przygotowania IBM WebSphere MQ , aby wprowadzić szczegóły nowego konta użytkownika.

Zmiana hasła konta użytkownika usługi produktu IBM WebSphere MQ Windows

O tym zadaniu

Aby zmienić hasło do lokalnego konta użytkownika usługiIBM WebSphere MQ Windows , wykonaj następujące kroki:

Procedura

1. Zidentyfikuj użytkownika, w którym usługa jest uruchomiona.
2. Zatrzymaj usługę IBMIBM WebSphere MQ z poziomu panelu Zarządzanie komputerem.
3. Zmień wymagane hasło w taki sam sposób, w jaki zmienisz hasło danej osoby.
4. Przejdź do właściwości usługi IBM WebSphere MQ z poziomu panelu Zarządzanie komputerem.
5. Wybierz stronę **Logowanie** .
6. Upewnij się, że podana nazwa konta jest zgodna z nazwą użytkownika, dla którego hasło zostało zmodyfikowane.
7. Wpisz hasło w polach **Hasło** i **Potwierdź hasło** , a następnie kliknij przycisk **OK**.

Usługa IBM WebSphere MQ Windows dla instalacji działającej pod kontem użytkownika domeny

O tym zadaniu

Jeśli usługa IBM WebSphere MQ Windows dla instalacji działa pod kontem użytkownika domeny, można również zmienić hasło dla konta w następujący sposób:

Procedura

1. Zmień hasło dla konta domenowego na kontrolerze domeny. Może być konieczne poproś administratora domeny o to, aby zrobił to dla Ciebie.
2. Wykonaj kroki, aby zmodyfikować stronę **Logowanie** dla usługi IBMIBM WebSphere MQ .

   Konto użytkownika, na którym działa usługa produktu IBM WebSphere MQ Windows , wykonuje dowolne komendy MQSC, które są wykonywane przez aplikacje interfejsu użytkownika lub są wykonywane automatycznie podczas uruchamiania systemu, zamykania systemu lub odtwarzania usługi. W związku z tym konto użytkownika musi mieć uprawnienia administracyjne w produkcie IBM WebSphere MQ . Domyślnie jest ona dodawana do lokalnej grupy **mqm** na serwerze. Jeśli przypisanie to zostanie usunięte, usługa IBM WebSphere MQ Windows nie będzie działać. Więcej informacji na temat praw użytkownika zawiera sekcja “Uprawnienia użytkownika wymagane w przypadku usługi IBM WebSphere MQ Windows” na stronie 68 .

   Jeśli wystąpi problem z zabezpieczeniem konta użytkownika, na którym działa usługa IBM WebSphere MQ Windows , komunikaty o błędach i opisy pojawią się w systemowym dzienniku zdarzeń.

Pojęcia pokrewne

“Korzystanie z katalogu Active Directory (tylko systemWindows)” na stronie 67
W niektórych konfiguracjach sieciowych, w których konta użytkowników są definiowane w kontrolerach domeny korzystających z opcji Active Directory, lokalne konto użytkownika IBM WebSphere MQ może nie mieć uprawnień wymaganych do wysyłania zapytań o członkostwo w grupie innych kont użytkowników domeny. Kreator przygotowania produktu IBM WebSphere MQ określa, czy jest to przypadek, przeprowadzając testy i pytając użytkownika o konfigurację sieci.

IBM WebSphere MQ koordynujący z programem Db2 jako menedżer zasobów

Jeśli menedżery kolejek są uruchamiane z poziomu IBM WebSphere MQ Explorerlub IBM WebSphere MQ V7i występują problemy podczas koordynowania produktu Db2, należy sprawdzić dzienniki błędów menedżera kolejek.

Sprawdź, czy w dziennikach błędów menedżera kolejek wystąpił błąd podobny do następującego:

```
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.
```

Objaśnienie: ID użytkownika (nazwa domyślna to MUSR_MQADMIN), który uruchamia procesIBM WebSphere MQ usługi `amqsvc.exe` , nadal działa z tokenem dostępu, który nie zawiera informacji o przypisach do grupy DB2USERS.

Rozwiąż: Po upewniu się, że ID użytkownika usługi IBM WebSphere MQ jest członkiem DB2USERS, należy użyć następującej sekwencji komend:

- zatrzymaj usługę.
- Zatrzymaj wszystkie inne procesy działające pod tym samym identyfikatorem użytkownika.
- ponownie uruchom te procesy.

Restartowanie komputera zapewniłoby wykonanie poprzednich kroków, ale nie jest to konieczne.

Rozszerzanie serwera IBM WebSphere MQ Explorer

IBM WebSphere MQ dla systemów Windowsi IBM WebSphere MQ for Linux (platformyx86 i x86-64) udostępniają interfejs administracyjny o nazwie IBM WebSphere MQ Explorer , który umożliwia wykonywanie zadań administracyjnych jako alternatywa dla używania komend sterujących lub komend MQSC.

Te informacje dotyczą tylko produktów WebSphere MQ for Windows i WebSphere MQ for Linux (tylko dla platformx86 i x86-64).

Eksplorator produktu IBM WebSphere MQ prezentuje informacje w stylu zgodnym ze strukturą środowiska Eclipse i innymi aplikacjami, które są obsługiwane przez środowisko Eclipse .

Dzięki rozszerzaniu programu IBM WebSphere MQ Explorer administratorzy systemu mają możliwość dostosowania programu WebSphere MQ Explorer w celu usprawnienia sposobu administrowania produktem WebSphere MQ.

Więcej informacji na ten temat zawiera sekcja *Rozszerzanie programu IBM WebSphere MQ Explorer* w dokumentacji produktu IBM WebSphere MQ Explorer.

Korzystanie z aplikacji paska zadań produktu IBM WebSphere MQ (tylko w systemie Windows)

Aplikacja paska zadań programu IBM WebSphere MQ wyświetla ikonę na pasku zadań systemu Windows na serwerze. Ikona udostępnia bieżący status produktu IBM WebSphere MQ oraz menu, z którego można wykonywać proste czynności.

W systemie Windowsikona WebSphere MQ znajduje się na pasku zadań serwera i jest nakładana za pomocą symbolu o kodzie kolorowym, który może mieć jeden z następujących znaczeń:

Zielony
Praca poprawnie; brak alertów w chwili obecnej

Niebieski
Nieokreślone; program WebSphere MQ uruchamia się lub wyłącza

Żółty
Alert; jedna lub więcej usług kończy się niepowodzeniem lub nie powiodło się

Aby wyświetlić menu, kliknij prawym przyciskiem myszy ikonę WebSphere MQ . Z poziomu menu można wykonać następujące czynności:

- Kliknij opcję **Otwórz** , aby otworzyć monitor alertów produktu WebSphere MQ .
- Kliknij przycisk **Wyjście** , aby wyjść z aplikacji WebSphere MQ Taskbar.
- Kliknij opcję **Eksplorator produktuWebSphere MQ** , aby uruchomić program IBM WebSphere MQ Explorer.
- Kliknij opcję **Zatrzymaj produkt WebSphere MQ** , aby zatrzymać produkt WebSphere MQ .
- Kliknij opcję **Informacje o produkcie WebSphere MQ** , aby wyświetlić informacje na temat monitora alertów WebSphere MQ

Aplikacja monitora alertów produktu IBM WebSphere MQ (tylko w systemie Windows)

Monitor alertów IBM WebSphere MQ to narzędzie do wykrywania błędów, które identyfikuje i rejestruje problemy z produktem IBM WebSphere MQ na komputerze lokalnym.

Monitor alertów wyświetla informacje na temat bieżącego statusu instalacji lokalnej serwera WebSphere MQ . Monitoruje on również interfejs ACPI (Advanced Configuration and Power Interface) systemu Windows i zapewnia wymuszanie ustawień ACPI.

Z poziomu monitora alertów produktu WebSphere MQ można:

- Bezpośredni dostęp do programu IBM WebSphere MQ Explorer
- Wyświetlanie informacji dotyczących wszystkich zaległych alertów
- Zamknij usługę WebSphere MQ na komputerze lokalnym.
- Kierowanie komunikatów o alertach za pośrednictwem sieci do konfigurowalnego konta użytkownika lub do stacji roboczej lub serwera Windows

Administrowanie lokalnymi obiektami IBM WebSphere MQ

W tej sekcji opisano sposób administrowania lokalnymi obiektami produktu IBM WebSphere MQ w celu obsługi aplikacji, które używają interfejsu kolejki komunikatów (Message Queue Interface-MQI). W tym kontekście lokalne administrowanie oznacza tworzenie, wyświetlanie, zmienianie, kopiowanie i usuwanie obiektów IBM WebSphere MQ .

Oprócz podejścia szczegółowo opisanego w tej sekcji można użyć eksploratora IBM WebSphere MQ do administrowania lokalnymi obiektami WebSphere MQ ; patrz “Administrowanie przy użyciu IBM WebSphere MQ Explorer” na stronie 58.

Ta sekcja zawiera następujące informacje:

- Programy aplikacji korzystające z interfejsu MQI
- “Wykonywanie lokalnych zadań administracyjnych za pomocą komend MQSC” na stronie 75
- “Praca z menedżerami kolejek” na stronie 84
- “Praca z kolejkami lokalnymi” na stronie 86
- “Praca z kolejkami aliasowymi” na stronie 91
- “Praca z kolejkami modelami” na stronie 93
- “Praca z usługami” na stronie 99
- “Zarządzanie obiektami w celu wyzwolenia” na stronie 106

Uruchamianie i zatrzymywanie menedżera kolejek

Ten temat zawiera wprowadzenie do zatrzymywania i uruchamiania menedżera kolejek.

Uruchamianie menedżera kolejek

Aby uruchomić menedżer kolejek, należy użyć komendy `strmqm` w następujący sposób:

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

W przypadku systemów WebSphere MQ for Windows i WebSphere MQ for Linux (platformyx86 i x86-64) menedżer kolejek można uruchomić w następujący sposób:

1. Otwórz program IBM WebSphere MQ Explorer.
2. Wybierz menedżera kolejek z widoku Navigator .
3. Kliknij opcję `Start`. Zostanie uruchomiony menedżer kolejek.

Jeśli uruchomienie menedżera kolejek trwa dłużej niż kilka sekund, program WebSphere MQ wysyła komunikaty informacyjne w sposób sporny z wyszczególniającym postępem uruchamiania.

Komenda `strmqm` nie zwraca kontroli, dopóki menedżer kolejek nie zostanie uruchomiony i będzie gotowy do akceptowania żądań połączeń.

Automatyczne uruchamianie menedżera kolejek

W produkcie WebSphere MQ for Windows menedżer kolejek można uruchamiać automatycznie, gdy system rozpoczyna korzystanie z programu WebSphere MQ Explorer. Więcej informacji na ten temat zawiera sekcja “Administrowanie przy użyciu IBM WebSphere MQ Explorer” na stronie 58.

Zatrzymywanie menedżera kolejek

Użyj komendy **endmqm** , aby zatrzymać menedżer kolejek.

Uwaga: Z instalacji powiązanej z menedżerem kolejek, z którym pracuje użytkownik, należy użyć komendy **endmqm** . Za pomocą komendy `dspmq -o installation` można dowiedzieć się, która instalacja menedżera kolejek jest powiązana.

Na przykład, aby zatrzymać menedżer kolejek o nazwie `QMB`, wprowadź następującą komendę:

```
endmqm QMB
```

W przypadku systemów WebSphere MQ for Windows i WebSphere MQ for Linux (platformyx86 i x86-64) można zatrzymać menedżer kolejek w następujący sposób:

1. Otwórz program IBM WebSphere MQ Explorer.
2. Wybierz menedżera kolejek z widoku Navigator .
3. Kliknij opcję `Stop...`. Zostanie wyświetlony panel Menedżer kolejek końcowych.
4. Wybierz opcję kontrolowaną lub natychmiast.
5. Kliknij opcję `OK`. Menedżer kolejek zostanie zatrzymany.

zamknięcie systemu z wygaszeniem

Domyślnie komenda **endmqm** wykonuje wygaszane zamknięcie określonego menedżera kolejek. Wykonanie tej operacji może potrwać pewien czas. Wygaszane zamknięcie czeka, aż wszystkie połączone aplikacje zostaną odłączone.

Użyj tego typu zamykania, aby powiadamiać aplikacje o zatrzymaniu. W przypadku wydania:

```
endmqm -c QMB
```

Użytkownik nie jest poinformowany, gdy wszystkie aplikacje zostały zatrzymane. (Komenda `endmqm -c QMB` jest równoważna komendzie `endmqm QMB`).

Jeśli jednak zostanie wydana:

```
endmqm -w QMB
```

Komenda oczekuje, aż wszystkie aplikacje zostaną zatrzymane, a menedżer kolejek zostanie zakończony.

natychmiastowe zatrzymanie

W przypadku natychmiastowego zamknięcia wszystkie bieżące wywołania MQI mogą być zakończone, ale żadne nowe wywołania nie powiodą się. Ten typ zamknięcia systemu nie czeka na rozłączenie aplikacji z menedżerem kolejek.

W przypadku natychmiastowego zamknięcia należy wpisać:

```
endmqm -i QMB
```

zamknięcie systemu z wywłaszczaniem

Uwaga: Nie należy używać tej metody, chyba że wszystkie inne próby zatrzymania menedżera kolejek za pomocą komendy **endmqm** nie powiodły się. Ta metoda może mieć nieprzewidywalne konsekwencje dla połączonych aplikacji.

Jeśli natychmiastowe zamknięcie systemu nie działa, należy użyć opcji zamknięcia *preemptive* , określając opcję `-p` . Na przykład:

```
endmqm -p QMB
```

Spowoduje to natychmiastowe zatrzymanie menedżera kolejek. Jeśli ta metoda nadal nie działa, należy zapoznać się z alternatywnym rozwiązaniem w sekcji “Ręczne zatrzymywanie menedżera kolejek” na stronie 73 .

Szczegółowy opis komendy **endmqm** i jej opcji można znaleźć w sekcji endmqm.

W przypadku problemów z zamknięciem menedżera kolejek

Problemy podczas zamykania menedżera kolejek są często spowodowane przez aplikacje. Na przykład, gdy aplikacje:

- Nie sprawdzaj poprawnie kodów powrotu MQI
- Nie żądaj powiadomienia o wygaszeniu
- Zakończ bez rozłączania z menedżerem kolejek (przez wywołanie wywołania `MQDISC`)

Jeśli problem wystąpi po zatrzymaniu menedżera kolejek, można przerwać wykonywanie komendy **endmqm** za pomocą kombinacji klawiszy Ctrl-C. Następnie można wprowadzić inną komendę **endmqm** , ale tym razem z flagą, która określa wymagany typ zamknięcia.

Ręczne zatrzymywanie menedżera kolejek

Jeśli standardowe metody zatrzymywania menedżerów kolejek nie powiodą się, należy spróbować opisanych tutaj metod.

Standardowy sposób zatrzymywania menedżerów kolejek jest używany przy użyciu komendy `endmqm` . Aby ręcznie zatrzymać menedżer kolejek, należy użyć jednej z procedur opisanych w tej sekcji. Szczegółowe informacje na temat wykonywania operacji na menedżerach kolejek za pomocą komend sterujących zawiera sekcja Tworzenie menedżerów kolejek i zarządzanie nimi.

Zatrzymywanie menedżerów kolejek w systemie Windows

Jak zakończyć procesy i usługę IBM WebSphere MQ , aby zatrzymać menedżery kolejek w programie IBM WebSphere MQ dla systemu Windows.

Aby zatrzymać menedżer kolejek działający w produkcie WebSphere MQ dla systemu Windows:

1. Lista nazw (identyfikatorów) procesów, które są uruchomione, za pomocą menedżera zadań Windows .

2. Zakończ procesy, używając menedżera zadań systemu Windows lub komendy **taskkill** , w następującej kolejności (jeśli są uruchomione):

AMQZMUC0	Menedżer procesu krytycznego
AMQZXMA0	Kontroler wykonywania
AMQZFUMA	Proces OAM
AMQZLAA0	Agenty LQM
AMQZLSA0	Agenty LQM
AMQZMUF0	Menedżer narzędzi
AMQZMGR0	Kontroler procesów
AMQZMUR0	Restartowalny menedżer procesu
AMQFQPUB	Proces publikowania subskrybowania
AMQFCXBA	Proces roboczy brokera
AMQRMPPA	Proces zestawiania procesów
AMQCRSTA	Proces zadania, który nie jest wątkiem
AMQCRS6B	Kanał odbiorczy LU62 i połączenie klienta
AMQRRMFA	Proces repozytorium (dla klastrów)
AMQZDMAA	Odroczony procesor komunikatów
AMQPCSEA	Serwer komend
RUNMQTRM	Wywołaj monitor wyzwalacza dla serwera
RUNMQDLQ	Wywołaj procedurę obsługi kolejki niedostarczonych komunikatów
RUNMQCHI	Proces inicjatora kanału
RUNMQLSR	Proces nasłuchiwania kanału
AMQXSSVN	Współużytkowane serwery pamięci
AMQZTRCN	Śledzenie

3. Zatrzymaj usługę WebSphere MQ za pomocą opcji **Narzędzia administracyjne** > **Usługi** w Panelu sterowania systemu Windows .
4. Jeśli próbowano wszystkich metod, a menedżer kolejek nie został zatrzymany, zrestartuj system.

Menedżer zadań systemu Windows i komenda **tasklist** udzieli ograniczonych informacji na temat zadań. Więcej informacji na temat określania, które procesy dotyczą konkretnego menedżera kolejek, można znaleźć przy użyciu narzędzia, takiego jak *Process Explorer* (procexp.exe), dostępnego do pobrania ze strony WWW firmy Microsoft pod adresem https://www.microsoft.com.

Zatrzymywanie menedżerów kolejek w systemach UNIX and Linux

Jak zakończyć procesy i usługę IBM WebSphere MQ , aby zatrzymać menedżery kolejek w produkcie IBM WebSphere MQ dla produktu UNIX and Linux. Metody opisane w tym miejscu można wypróbować, jeśli standardowe metody zatrzymywania i usuwania menedżerów kolejek nie powiodą się.

Aby zatrzymać menedżer kolejek działający w produkcie WebSphere MQ dla systemów UNIX and Linux :

1. Znajdź identyfikatory procesów dla programów menedżera kolejek, które nadal działają, za pomocą komendy `ps` . Na przykład, jeśli menedżer kolejek ma nazwę `QMNAME`, użyj następującej komendy:

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

2. Zakończ wszystkie procesy menedżera kolejek, które nadal działają. Należy użyć komendy `kill` , określając identyfikatory procesów wykryte za pomocą komendy `ps` .

Zakończ procesy w następującej kolejności:

amqzmuc0	Menedżer procesu krytycznego
amqzxma0	Kontroler wykonywania
amqzfuma	Proces OAM
amqzlaa0	Agenty LQM
amqzlsa0	Agenty LQM
amqzmuf0	Menedżer narzędzi
amqzmur0	Restartowalny menedżer procesu
amqzmgr0	Kontroler procesów
amqfqpub	Proces publikowania subskrybowania
amqfcxba	Proces roboczy brokera
amqrmppa	Proces zestawiania procesów
amqcrsta	Proces zadania, który nie jest wątkiem
amqcrs6b	Kanał odbiorczy LU62 i połączenie klienta
amqrrmfa	Proces repozytorium (dla klastrów)
amqzdmaa	Odroczony procesor komunikatów
amqpcsea	Serwer komend
runmqtrm	Wywołaj monitor wyzwalacza dla serwera
runmqdlq	Wywołaj procedurę obsługi kolejki niedostarczonych komunikatów
runmqchi	Proces inicjatora kanału
runmqlsr	Proces nasłuchiwania kanału

Uwaga: Komendy **`kill -9`** można użyć do zakończenia procesów, które nie zostały zatrzymane.

Jeśli menedżer kolejek zostanie zatrzymany ręcznie, mogą zostać pobrane pliki FFST, a pliki FDC umieszczone w programie `/var/mqm/errors`. Nie należy traktować tego jako defektu w menedżerze kolejek.

Menedżer kolejek zostanie zrestartowany normalnie, nawet po jego zatrzymaniu przy użyciu tej metody.

Wykonywanie lokalnych zadań administracyjnych za pomocą komend MQSC

Ta sekcja zawiera wprowadzenie do komend MQSC i instruuje, jak używać ich dla niektórych typowych zadań.

Jeśli używany jest produkt IBM WebSphere MQ dla systemu Windows lub IBM WebSphere MQ dla systemu Linux (platformyx86 i x86-64), można również wykonać operacje opisane w tej sekcji za pomocą programu IBM WebSphere MQ Explorer. Więcej informacji zawiera sekcja “Administrowanie przy użyciu IBM WebSphere MQ Explorer” na stronie 58.

Za pomocą komend MQSC można zarządzać obiektami menedżera kolejek, w tym samym menedżerem kolejek, kolejkami, definicjami procesów, kanałami, kanałami połączeń klientów, obiektami nasłuchiwania, usługami, listami nazw, klastrami i obiektami informacji uwierzytelniających. Ta sekcja dotyczy menedżerów kolejek, kolejek i definicji procesów. Informacje na temat administrowania kanałami, kanałami połączeń klienta i obiektami nasłuchiwania zawiera sekcja Obiekty . Więcej informacji na temat wszystkich komend MQSC służących do zarządzania obiektami menedżera kolejek zawiera sekcja “Komendy skryptowe (MQSC)” na stronie 76.

Komendy MQSC są emitowane do menedżera kolejek za pomocą komendy `runmqsc` . Szczegółowe informacje na temat tej komendy można znaleźć w sekcji runmqsc. Można to zrobić interaktywnie,

wydając komendy za pomocą klawiatury lub można przekierować standardowe urządzenie wejściowe (`stdin`), aby uruchomić sekwencję komend z pliku tekstowego ASCII. W obu przypadkach format komend jest taki sam. (Informacje na temat uruchamiania komend z pliku tekstowego można znaleźć w sekcji “Uruchamianie komend MQSC z plików tekstowych” na stronie 79).

Komendę `runmqsc` można uruchomić na trzy sposoby, w zależności od opcji ustawionych w komendzie:

- Sprawdź, czy komenda nie została uruchomiona, gdzie komendy MQSC są weryfikowane w lokalnym menedżerze kolejek, ale nie są wykonywane.
- Uruchom komendę w lokalnym menedżerze kolejek, w którym komendy MQSC są uruchamiane w lokalnym menedżerze kolejek.
- Uruchom komendę w zdalnym menedżerze kolejek, w którym komendy MQSC są uruchamiane w zdalnym menedżerze kolejek.

Można również uruchomić komendę, po której następuje znak zapytania, aby wyświetlić składnię.

Atrybuty obiektów określone w komendach MQSC są wyświetlane w tej sekcji wielkimi literami (na przykład RQMNAME), chociaż nie są one rozróżniane wielkości liter. Nazwy atrybutów komend MQSC są ograniczone do ośmiu znaków. Komendy MQSC są dostępne na innych platformach, w tym w plikach IBM i i z/OS.

Komendy MQSC są podsumowywane w kolekcji tematów w sekcji MQSC reference (Skorowidz komend MQSC).

Komendy skryptowe (MQSC)

Komendy MQSC udostępniają jednolitą metodę wydawania komend do odczytu przez człowieka na platformach WebSphere MQ . Informacje na temat komend programu *programmable command format* (PCF) można znaleźć w sekcji “Wprowadzenie do formatów komend programowalnych” na stronie 9.

Ogólny format komend jest wyświetlany w sekcji Komendy MQSC.

Podczas korzystania z komend MQSC należy przestrzegać następujących reguł:

- Każda komenda zaczyna się od parametru podstawowego (czasownika), a po nim następuje dodatkowy parametr (rzeczownik). Następnie następuje po nim nazwa lub ogólna nazwa obiektu (w nawiasach), jeśli istnieje jedna, która znajduje się w większości komend. W związku z tym, parametry mogą zwykle występować w dowolnej kolejności; jeśli parametr ma odpowiednią wartość, wartość musi być wykonywana bezpośrednio po parametrze, do którego się odnosi.
- Słowa kluczowe, nawiasy i wartości mogą być rozdzielane dowolną liczbą odstępów i przecinków. Przecinek wyświetlany na diagramach składniowych zawsze może być zastąpiony przez co najmniej jeden znak odstępu. Musi istnieć co najmniej jedno puste pole bezpośrednio poprzedzające każdy parametr (po parametrze podstawowym).
- Na początku lub na końcu komendy może wystąpić dowolna liczba odstępów oraz między parametrami, interpunkcją i wartościami. Na przykład poprawna jest następująca komenda:

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

  Odstępy w obrębie pary znaków cudzysłowu są znaczące.
- Dodatkowe przecinki mogą pojawiać się w dowolnym miejscu, gdzie dozwolone są odstępy i są traktowane tak, jakby były odstępami (chyba, że są one oczywiście wewnątrz łańcuchów ujętych w cudzysłów).
- Powtarzające się parametry nie są dozwolone. Powtarzanie parametru z jego wersją "NO", tak jak w REPLACE NOREPLACE, również nie jest dozwolone.
- Łańcuchy, które zawierają spacje, małe litery lub znaki specjalne inne niż:
  - Kropka (.)
  - Prawy ukośnik (/)
  - Podkreślenie (_)

– Znak procentu (%)

muszą być ujęte w apostrofach, o ile nie są:

– Wartości ogólne kończące się gwiazdką
– Pojedyncza gwiazdka (na przykład TRACE (*))
– Specyfikacja zakresu zawierająca dwukropek (na przykład: KLASA (01:03))

Jeśli sam łańcuch zawiera pojedynczy cudzysłów, pojedynczy cudzysłów jest reprezentowany przez dwa apostrofowe znaki cudzysłowu.Małe litery, które nie są ujęte w cudzysłów, są składane wielkimi literami.

- Na platformach innych niż z/OSłańcuch zawierający znaki bez znaków (czyli dwa pojedyncze cudzysłowy bez spacji między) jest interpretowany jako spacja ujęta w znaki pojedynczego cudzysłowu, czyli interpretowana w ten sam sposób co (''). Wyjątkiem jest ten wyjątek, jeśli używany atrybut ma jedną z następujących wartości:

  – TOPICSTR
  – SUB
  – USERDATA
  – SELECTOR

  wtedy dwa pojedyncze cudzysłowy bez spacji są interpretowane jako łańcuch o zerowej długości.
- W produkcie v7.0wszystkie odstępy końcowe w tych atrybutach łańcuchowych, które są oparte na typach MQCHARV, takie jak SELECTOR, dane użytkownika podrzędnego, są traktowane jako znaczące, co oznacza, że 'abc' nie równa się "abc".
- Lewy nawias, po którym następuje prawy nawias, bez istotnych informacji pomiędzy, na przykład

  ```
  NAME ( )
  ```

  jest niepoprawna, z wyjątkiem przypadków, w których wyraźnie zaznaczono.
- W słowach kluczowych nie jest rozróżniana wielkość liter: AltER, alter, and ALTER są akceptowalne.Wszystko, co nie jest zawarte w cudzysłowie, jest składane na wielkie litery.
- Synonimy są zdefiniowane dla niektórych parametrów.Na przykład, DEF jest zawsze synonimem dla DEFINE, więc DEF QLOCAL jest ważne. Synonimy nie są jednak tylko łańcuchami minimalnymi; DEFI nie jest poprawnym synonimem dla definicji DEFINE.

  Uwaga: Dla parametru DELETE nie ma synonimu. Ma to na celu uniknięcie przypadkowego usunięcia obiektów podczas korzystania z DEF, czyli synonimu dla DEFINE.

Przegląd informacji na temat używania komend MQSC do administrowania produktem IBM WebSphere MQznajduje się w sekcji “Wykonywanie lokalnych zadań administracyjnych za pomocą komend MQSC” na stronie 75.

Komendy MQSC używają określonych znaków specjalnych do określenia określonych znaczeń. Więcej informacji na temat tych znaków specjalnych oraz sposobu ich używania zawiera sekcja Znaki ze specjalnymi znaczeniami.

Aby dowiedzieć się, w jaki sposób można budować skrypty za pomocą komend MQSC, należy zapoznać się z Budowanie skryptów komend.

Pełną listę komend MQSC można znaleźć w sekcji Komendy MQSC.

Zadania pokrewne
Budowanie skryptów komend

Nazwy obiektów WebSphere MQ

Informacje na temat używania nazw obiektów w komendach MQSC.

W przykładach używamy niektórych długich nazw obiektów. Jest to pomocne w określeniu typu obiektu, z którym użytkownik ma do czynienia.

Podczas wydawania komend MQSC konieczne jest podanie tylko nazwy lokalnej kolejki. W naszych przykładach używamy nazw kolejek takich jak:

```
ORANGE.LOCAL.QUEUE
```

Część nazwy `LOCAL.QUEUE` ma na celu zilustrować, że ta kolejka jest kolejką lokalną. Jest to ***nie*** wymagane w przypadku nazw kolejek lokalnych.

Nazwa `saturn.queue.manager` jest również używana jako nazwa menedżera kolejek. Część nazwy `queue.manager` ma na celu zilustrować, że ten obiekt jest menedżerem kolejek. *Nie* jest to wymagane w przypadku nazw menedżerów kolejek w ogóle.

Rozróżnianie wielkości liter w komendach MQSC

Komendy MQSC, w tym ich atrybuty, mogą być zapisywane wielkimi lub małymi literami. Nazwy obiektów w komendach MQSC są składane na wielkie litery (to znaczy w kolejce i w kolejce nie są rozróżniane), chyba że nazwy są ujęte w znaki pojedynczego cudzysłowu. Jeśli znaki cudzysłowu nie są używane, obiekt jest przetwarzany z nazwą zapisanej wielkimi literami. Więcej informacji na ten temat zawiera sekcjaInformacje dodatkowe dotyczące komend MQSC .

Wywołanie komendy `runmqsc` , wspólne dla wszystkich komend sterujących WebSphere MQ , jest rozróżniana w niektórych środowiskach WebSphere MQ . Więcej informacji na ten temat zawiera sekcja Używanie komend sterujących .

Standardowe wejście i wyjście

Standardowe urządzenie wejściowe, zwane również `stdin`, jest urządzeniem, z którego pobierane są dane wejściowe do systemu. Zwykle jest to klawiatura, ale można określić, że dane wejściowe mają pochodzić z portu szeregowego lub z pliku na dysku, np. *Standardowe urządzenie wyjściowe*, zwane również `stdout`, jest urządzeniem, do którego wysyłane są dane wyjściowe z systemu. Zwykle jest to ekran, ale można przekierować dane wyjściowe do portu szeregowego lub pliku.

W komendach systemu operacyjnego oraz w komendach sterujących WebSphere MQ < operator przekierowuje dane wejściowe. Jeśli po tym operatorze występuje nazwa pliku, dane wejściowe są pobierane z pliku. Podobnie, > operator przekierowuje dane wyjściowe; jeśli po nim następuje nazwa pliku, dane wyjściowe są kierowane do tego pliku.

Interaktywna obsługa komend MQSC

Komendy MQSC można używać interaktywnie, korzystając z okna komend lub powłoki.

Aby używać komend MQSC interaktywnie, otwórz okno komend lub powłokę i wpisz:

```
runmqsc
```

W tej komendzie nie określono nazwy menedżera kolejek, dlatego komendy MQSC są przetwarzane przez domyślny menedżer kolejek. Jeśli ma być używany inny menedżer kolejek, należy podać nazwę menedżera kolejek w komendzie **runmqsc** . Na przykład, aby uruchomić komendy MQSC dla menedżera kolejek `jupiter.queue.manager`, należy użyć komendy:

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

Po wykonaniu tej czynności wszystkie typy komend MQSC używane przez użytkownika są przetwarzane przez ten menedżer kolejek, zakładając, że jest on w tym samym węźle i jest już uruchomiony.

Teraz można wpisać dowolne komendy MQSC, zgodnie z wymaganiami. Na przykład spróbuj wykonać następujące czynności:

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

W przypadku komend, które mają zbyt wiele parametrów, aby zmieścić się w jednym wierszu, należy użyć znaków kontynuacji w celu wskazania, że komenda jest kontynuowana w następującym wierszu:

- Znak minus (-) wskazuje, że komenda ma być kontynuowana od początku następnego wiersza.
- Znak plus (+) wskazuje, że komenda ma być kontynuowana od pierwszego niepustego znaku w następnym wierszu.

Wejście komendy kończy się znakiem finałowym niepustej linii, która nie jest znakiem kontynuacji. Dane wejściowe komendy można również zakończyć jawnie, wprowadzając średnik (;). (Jest to szczególnie przydatne w przypadku przypadkowego wprowadzenia znaku kontynuacji na końcu ostatniego wiersza wprowadzania komendy).

Informacje zwrotne z komend MQSC

Po wydaniu komend MQSC menedżer kolejek zwraca komunikaty operatora, które potwierdzają działania użytkownika, lub informacje o błędach, które zostały wykonane. Na przykład:

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

Ten komunikat potwierdza, że kolejka została utworzona.

```
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
```

Ten komunikat wskazuje, że wystąpił błąd składniowy.

Komunikaty te są wysyłane do standardowego urządzenia wyjściowego. Jeśli komenda nie została poprawnie wprowadzona, należy zapoznać się z poprawką składni w sekcji Informacje dodatkowe dotyczące komend MQSC .

Zakończenie interaktywnego wprowadzania komend MQSC

Aby zakończyć pracę z komendami MQSC, należy wprowadzić komendę END.

Alternatywnie można użyć znaku EOF dla danego systemu operacyjnego.

Uruchamianie komend MQSC z plików tekstowych

Uruchamianie komend MQSC interaktywnie jest odpowiednie do szybkich testów, ale jeśli użytkownik ma bardzo długie komendy lub wielokrotnie używa określonej sekwencji komend, należy rozważyć przekierowanie programu `stdin` z pliku tekstowego.

“Standardowe wejście i wyjście” na stronie 78 zawiera informacje na temat produktów `stdin` i `stdout`. Aby przekierować program `stdin` z pliku tekstowego, należy najpierw utworzyć plik tekstowy zawierający komendy MQSC, korzystając z zwykłego edytora tekstu. Jeśli używana jest komenda `runmqsc` , należy

użyć operatorów przekierowania. Na przykład następująca komenda uruchamia sekwencję komend zawartych w pliku tekstowym `myprog.in`:

```
runmqsc < myprog.in
```

Podobnie, można również przekierować dane wyjściowe do pliku. Plik zawierający komendy MQSC dla danych wejściowych nosi nazwę *pliku komend MQSC*. Plik wyjściowy zawierający odpowiedzi z menedżera kolejek jest nazywany *plikiem wyjściowym*.

Aby przekierować zarówno produkty `stdin` , jak i `stdout` w komendzie `runmqsc` , należy użyć następującej formy komendy:

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

Ta komenda wywołuje komendy MQSC zawarte w pliku komend MQSC `myprog.in`. Ponieważ nie określiliśmy nazwy menedżera kolejek, komendy MQSC są uruchamiane dla domyślnego menedżera kolejek. Dane wyjściowe są wysyłane do pliku tekstowego `myprog.out`. Rysunek 12 na stronie 80 przedstawia wyodrębnienie z pliku komend MQSC `myprog.in` , a Rysunek 13 na stronie 81 przedstawia odpowiedni wyciąg danych wyjściowych w produkcie `myprog.out`.

Aby przekierować `stdin` i `stdout` w komendzie `runmqsc` , dla menedżera kolejek (`saturn.queue.manager`), który nie jest domyślny, należy użyć następującej postaci komendy:

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

Pliki komend MQSC

Komendy MQSC są zapisywane w postaci czytelnej dla użytkownika, to znaczy w tekście ASCII. Rysunek 12 na stronie 80 Jest to wyciąg z pliku komend MQSC, który zawiera komendę MQSC (DEFINE QLOCAL) z jej atrybutami. Informacje dodatkowe dotyczące komend MQSC zawiera opis każdej komendy MQSC i jej składni.

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

Rysunek 12. Wyodrębnij z pliku komend MQSC

W celu zapewnienia przenośności między środowiskami WebSphere MQ należy ograniczyć długość linii w plikach komend MQSC do 72 znaków. Znak plus wskazuje, że komenda jest kontynuowana w następnym wierszu.

Raporty komend MQSC

Komenda **runmqsc** zwraca *raport*, który jest wysyłany do programu `stdout`. Raport zawiera:

- Nagłówek identyfikujący komendy MQSC jako źródło raportu:

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

  Gdzie `jupiter.queue.manager` to nazwa menedżera kolejek.
- Opcjonalna numerowana lista wydanych komend MQSC. Domyślnie tekst danych wejściowych jest odbity na wyjściu. W tym wyniku każda komenda jest poprzedzona numerem kolejnym, jak to pokazano na rysunku Rysunek 13 na stronie 81. Można jednak użyć opcji -e w komendzie `runmqsc` , aby wyłączyć dane wyjściowe.
- Komunikat o błędzie składni dla wszystkich komend, które mogą być błędnie używane.
- *Komunikat operatora* wskazujący wynik działania każdej komendy. Na przykład komunikat operatora dla pomyślnego zakończenia komendy DEFINE QLOCAL jest następujący:

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

- Inne komunikaty wynikające z błędów ogólnych podczas uruchamiania pliku skryptowego.
- Krótkie statystyczne podsumowanie raportu, wskazujące liczbę odczytanych komend, liczbę komend z błędami składniową oraz liczbę komend, które nie mogły zostać przetworzone.

  Uwaga: Menedżer kolejek próbuje przetworzyć tylko te komendy, które nie zawierają błędów składniowych.

```
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.
      :
 .
 .
```

Rysunek 13. Wyodrębnij z pliku raportu komend MQSC

Uruchamianie dostarczonych plików komend MQSC

Z produktem WebSphere MQdostarczane są następujące pliki komend MQSC:

amqscos0.tst
Definicje obiektów używanych przez programy przykładowe.

amqscic0.tst
Definicje kolejek dla transakcji CICS .

W produkcie WebSphere MQ for Windowspliki te znajdują się w katalogu *MQ_INSTALLATION_PATH*`\tools\mqsc\samples`. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

W systemach UNIX and Linux te pliki znajdują się w katalogu *MQ_INSTALLATION_PATH*/samp. *MQ_INSTALLATION_PATH* reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

Komenda, która je uruchamia, jest następująca:

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

Używanie komendy runmqsc do weryfikowania komend

Komendy runmqsc można użyć do sprawdzenia komend MQSC w lokalnym menedżerze kolejek bez ich rzeczywistego uruchamiania. Aby to zrobić, należy ustawić flagę -v w komendzie runmqsc , na przykład:

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

Po wywołaniu komendy runmqsc dla pliku komend MQSC, menedżer kolejek weryfikuje każdą komendę i zwraca raport bez wykonywania komend MQSC. Pozwala to na sprawdzenie składni komend w pliku komend. Jest to szczególnie ważne w przypadku, gdy:

- Uruchamianie dużej liczby komend z pliku komend.
- Korzystanie z pliku komend MQSC wiele razy.

Zwrócony raport jest podobny do przedstawionego w sekcji Rysunek 13 na stronie 81.

Tej metody nie można używać do zdalnego sprawdzania komend MQSC. Na przykład, jeśli podejmowana jest próba wykonania tej komendy:

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

Opcja -w, która wskazuje, że menedżer kolejek jest zdalny, jest ignorowana, a komenda jest uruchamiana lokalnie w trybie weryfikacji. Wartość 30 to liczba sekund, przez jaką produkt WebSphere MQ oczekuje na odpowiedzi ze zdalnego menedżera kolejek.

Uruchamianie komend MQSC z plików wsadowych

Jeśli użytkownik ma bardzo długie komendy lub wielokrotnie używa określonej sekwencji komend, należy rozważyć przekierowanie produktu stdin z pliku wsadowego.

Aby przekierować program stdin z pliku wsadowego, należy najpierw utworzyć plik wsadowy zawierający komendy MQSC, korzystając z zwykłego edytora tekstu. Jeśli używana jest komenda runmqsc , należy użyć operatorów przekierowania. Poniższy przykład:

1. Tworzy menedżera kolejek testowych, TESTQM
2. Tworzy zgodny zestaw CLNTCONN i zestaw nasłuchujący, aby użyć portu TCP/IP 1600
3. Tworzy kolejkę testową TESTQ
4. Umieszcza komunikat w kolejce przy użyciu przykładowego programu 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
```

Rysunek 14. Przykładowy skrypt do uruchamiania komend MQSC z pliku wsadowego

Rozwiązywanie problemów za pomocą komend MQSC

Jeśli nie można pobrać komend MQSC do uruchomienia, należy zapoznać się z informacjami w tym temacie, aby sprawdzić, czy jakiekolwiek z tych typowych problemów dotyczą użytkownika. Nie zawsze jest oczywiste, czym jest problem, gdy czytasz błąd generowany przez komendę.

Jeśli nie można pobrać komend MQSC do uruchomienia, należy użyć poniższych informacji, aby sprawdzić, czy jakiekolwiek z tych typowych problemów mają zastosowanie. Nie zawsze jest oczywiste, czym jest problem podczas odczytywania wygenerowanego błędu.

Jeśli używana jest komenda `runmqsc` , należy pamiętać o następujących przypadkach:

- Użyj operatora **<** , aby przekierować dane wejściowe z pliku. Jeśli ten operator zostanie pominięty, menedżer kolejek interpretuje nazwę pliku jako nazwę menedżera kolejek i wysyła następujący komunikat o błędzie:

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

- Jeśli dane wyjściowe są przekierowane do pliku, należy użyć operatora przekierowania **>** . Domyślnie plik jest umieszczany w bieżącym katalogu roboczym w momencie wywołania programu `runmqsc` . Podaj pełną nazwę pliku, aby wysłać dane wyjściowe do określonego pliku i katalogu.
- Sprawdź, czy został utworzony menedżer kolejek, który ma uruchamiać te komendy, za pomocą następującej komendy w celu wyświetlenia wszystkich menedżerów kolejek:

  ```
  dspmq
  ```

- Menedżer kolejek musi być uruchomiony. Jeśli tak nie jest, uruchom go; (patrz Uruchamianie menedżera kolejek). W przypadku próby uruchomienia menedżera kolejek, który jest już uruchomiony, wyświetlany jest komunikat o błędzie.
- Podaj nazwę menedżera kolejek w komendzie `runmqsc` , jeśli nie zdefiniowano domyślnego menedżera kolejek, lub jeśli wystąpi ten błąd:

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

- Nie można określić komendy MQSC jako parametru komendy `runmqsc` . Na przykład nie jest to poprawne:

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

- Nie można wprowadzić komend MQSC przed wydaniem komendy `runmqsc` .
- Nie można uruchamiać komend sterujących z `runmqsc`. Na przykład nie można uruchomić komendy `strmqm` w celu uruchomienia menedżera kolejek w czasie, gdy komendy MQSC są uruchamiane interaktywnie. W takim przypadku zostaną wyświetlone komunikaty o błędach podobne do poniższych:

```
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
```

Praca z menedżerami kolejek

Przykłady komend MQSC, których można używać do wyświetlania lub modyfikowania atrybutów menedżera kolejek.

Wyświetlanie atrybutów menedżera kolejek

Aby wyświetlić atrybuty menedżera kolejek określonego w komendzie **runmqsc** , należy użyć następującej komendy MQSC:

```
DISPLAY QMGR
```

Typowe dane wyjściowe tej komendy są przedstawione w sekcji Rysunek 15 na stronie 85

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

Rysunek 15. Typowe dane wyjściowe komendy DISPLAY QMGR

Parametr ALL jest wartością domyślną w komendzie DISPLAY QMGR . Wyświetla on wszystkie atrybuty menedżera kolejek. W szczególności dane wyjściowe informują o domyślnej nazwie menedżera kolejek, nazwie kolejki niedostarczonych komunikatów oraz o nazwie kolejki komend.

Można sprawdzić, czy te kolejki istnieją, wprowadzając komendę:

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

Spowoduje to wyświetlenie listy kolejek zgodnych z łodyg SYSTEM.*. Nawiasy są wymagane.

Zmiana atrybutów menedżera kolejek

Aby zmienić atrybuty menedżera kolejek określonego w komendzie **runmqsc** , należy użyć komendy MQSC ALTER QMGR, określając atrybuty i wartości, które mają zostać zmienione. Na przykład można użyć następujących komend, aby zmienić atrybuty produktu jupiter.queue.manager:

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

Komenda ALTER QMGR służy do zmiany używanej kolejki niedostarczonych komunikatów, a także umożliwia zahamowanie zdarzeń.

Odsyłacze pokrewne

Atrybuty dla menedżera kolejek

Praca z kolejkami lokalnymi

Ta sekcja zawiera przykłady niektórych komend MQSC, których można używać do zarządzania kolejkami lokalnymi, modelowymi i aliasowymi.

Szczegółowe informacje na temat tych komend można znaleźć w publikacji Informacje dodatkowe dotyczące komend MQSC .

Definiowanie kolejki lokalnej

W przypadku aplikacji lokalny menedżer kolejek jest menedżerem kolejek, z którym połączona jest aplikacja. Kolejki zarządzane przez lokalny menedżer kolejek są lokalne względem tego menedżera kolejek.

Aby utworzyć kolejkę lokalną, należy użyć komendy MQSC DEFINE QLOCAL. Można również użyć wartości domyślnej zdefiniowanej w domyślnej definicji kolejki lokalnej lub można zmodyfikować charakterystykę kolejki z domyślnej lokalnej kolejki.

Uwaga: Domyślna kolejka lokalna ma nazwę SYSTEM.DEFAULT.LOCAL.QUEUE i została utworzona w instalacji systemu.

Na przykład komenda DEFINE QLOCAL, która jest zgodna, definiuje kolejkę o nazwie ORANGE.LOCAL.QUEUE o następujących parametrach:

- Jest on włączony dla operacji pobierania, włączony dla operacji put i działa w oparciu o kolejność priorytetów.
- Jest to kolejka *normalna* . Nie jest to kolejka inicjujący ani kolejka transmisji, a nie generuje komunikatów wyzwalacza.
- Maksymalna głębokość kolejki to 5000 komunikatów. Maksymalna długość komunikatu to 4194304 bajtów.

```
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);
```

Uwaga:

1. Z wyjątkiem wartości dla opisu, wszystkie wartości atrybutów są wartościami domyślnymi. Pokazaliśmy je tutaj dla celów ilustracyjnych. Można je pominąć, jeśli użytkownik ma pewność, że wartości domyślne są takie, które mają zostać zmienione lub które nie zostały zmienione. Patrz także “Wyświetlanie domyślnych atrybutów obiektów” na stronie 87.
2. UŻYCIE (NORMALNE) wskazuje, że ta kolejka nie jest kolejką transmisji.
3. Jeśli w tym samym menedżerze kolejek znajduje się już kolejka lokalna o nazwie ORANGE.LOCAL.QUEUE. Ta komenda nie powiedzie się. Użyj atrybutu REPLACE, jeśli chcesz nadpisać istniejącą definicję kolejki, ale patrz także “Zmiana atrybutów kolejki lokalnej” na stronie 88.

Definiowanie kolejki niedostarczonych komunikatów

Każdy menedżer kolejek musi mieć kolejkę lokalną, która ma być używana jako kolejka niedostarczonych komunikatów, dzięki czemu komunikaty, których nie można dostarczyć do ich poprawnego miejsca docelowego, mogą zostać zapisane w celu późniejszego pobrania. Należy poinformować menedżera kolejek o kolejce niedostarczonych komunikatów.

Aby poinformować menedżera kolejek o kolejce niedostarczonych komunikatów, należy podać nazwę kolejki niedostarczonych komunikatów w komendzie `crtmqm` (na przykład`crtmqm -u DEAD.LETTER.QUEUE`) lub za pomocą atrybutu DEADQ komendy ALTER QMGR, aby określić jedną

z nich później. Przed użyciem kolejki niedostarczonych komunikatów należy zdefiniować kolejkę niedostarczonych komunikatów.

Przykładowa kolejka niedostarczonych komunikatów o nazwie SYSTEM.DEAD.LETTER.QUEUE jest dostępna razem z produktem. Ta kolejka jest tworzona automatycznie podczas tworzenia menedżera kolejek. Jeśli jest to wymagane, można zmodyfikować tę definicję i zmienić jej nazwę.

Kolejka niedostarczonych komunikatów nie ma specjalnych wymagań, z wyjątkiem sytuacji, gdy:

- Musi to być kolejka lokalna
- Atrybut MAXMSGL (maksymalna długość komunikatu) musi włączyć kolejkę w celu uwzględnienia największych komunikatów, które menedżer kolejek musi obsłużyć **plus** wielkość nagłówka niedostarczonych komunikatów (MQDLH)

Produkt WebSphere MQ udostępnia procedurę obsługi kolejki niedostarczonych komunikatów, która umożliwia określenie sposobu, w jaki komunikaty znalezione w kolejce niedostarczonych komunikatów mają być przetwarzane lub usuwane. Więcej informacji na ten temat zawiera sekcja Obsługa niedostarczonych komunikatów przy użyciu procedury obsługi kolejki niedostarczonych komunikatów w produkcie WebSphere MQ .

Wyświetlanie domyślnych atrybutów obiektów

Za pomocą komendy DISPLAY QUEUE można wyświetlić atrybuty, które zostały pobrane z obiektu domyślnego, gdy zdefiniowano obiekt WebSphere MQ .

Podczas definiowania obiektu WebSphere MQ przyjmuje on wszystkie atrybuty, które nie zostały określone na podstawie obiektu domyślnego. Na przykład podczas definiowania kolejki lokalnej kolejka dziedziczy wszystkie atrybuty, które są pomijane w definicji z domyślnej kolejki lokalnej, która jest nazywana SYSTEM.DEFAULT.LOCAL.QUEUE. Aby dokładnie sprawdzić, jakie są te atrybuty, należy użyć następującej komendy:

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

Składnia tej komendy różni się od składni odpowiadającej komendzie DEFINE. W komendzie DISPLAY można podać tylko nazwę kolejki, podczas gdy w komendzie DEFINE należy określić typ kolejki, czyli QLOCAL, QALIAS, QMODEL lub QREMOTE.

Można selektywnie wyświetlać atrybuty, określając je pojedynczo. Na przykład:

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

Ta komenda wyświetla trzy określone atrybuty w następujący sposób:

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

CURDEPTH to bieżąca głębokość kolejki, to znaczy liczba komunikatów w kolejce. Jest to przydatny atrybut do wyświetlenia, ponieważ monitorując głębokość kolejki, można się upewnić, że kolejka nie jest pełna.

Kopiowanie definicji kolejki lokalnej

Definicję kolejki można skopiować za pomocą atrybutu LIKE w komendzie DEFINE.

Na przykład:

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

Ta komenda tworzy kolejkę o takich samych atrybutach, jak w przypadku oryginalnej kolejki ORANGE.LOCAL.QUEUE, a nie w domyślnej lokalnej kolejce systemowej. Wprowadź nazwę kolejki, która ma zostać skopiowana ***dokładnie*** , ponieważ została ona wprowadzona podczas tworzenia kolejki. Jeśli nazwa zawiera małe litery, należy ująć ją w znaki pojedynczego cudzysłowu.

Można również użyć tego formularza komendy DEFINE w celu skopiowania definicji kolejki, ale należy zastąpić jedną lub więcej zmian w atrybutach oryginału. Na przykład:

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

Ta komenda kopiuje atrybuty kolejki ORANGE.LOCAL.QUEUE do kolejki THIRD.QUEUE, ale określa, że maksymalna długość komunikatu w nowej kolejce ma wynosić 1024 bajty, a nie 4194304.

Uwaga:

1. Jeśli atrybut LIKE jest używany w komendzie DEFINE, kopiowane są tylko atrybuty kolejki. Nie kopiujesz komunikatów w kolejce.
2. Jeśli definiujesz kolejkę lokalną, bez określania LIKE, jest ona taka sama, jak zdefiniowana w definicji DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE).

Zmiana atrybutów kolejki lokalnej

Atrybuty kolejki można zmienić na dwa sposoby, używając komendy ALTER QLOCAL lub komendy DEFINE QLOCAL z atrybutem REPLACE.

W programie “Definiowanie kolejki lokalnej” na stronie 86jest to kolejka o nazwie ORANGE.LOCAL.QUEUE została zdefiniowana. Przypuśćmy na przykład, że chcesz zmniejszyć maksymalną długość komunikatu w tej kolejce do 10 000 bajtów.

- Za pomocą komendy ALTER:

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

  Ta komenda zmienia jeden atrybut o maksymalną długość komunikatu. Wszystkie pozostałe atrybuty pozostają takie same.
- W przypadku użycia komendy DEFINE z opcją REPLACE, na przykład:

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

  Ta komenda zmienia nie tylko maksymalną długość komunikatu, ale także wszystkie inne atrybuty, które mają podane wartości domyślne. Kolejka jest teraz włączona, podczas gdy poprzednio została ona zablokowana. Włączona opcja put jest wartością domyślną, określoną w kolejce SYSTEM.DEFAULT.LOCAL.QUEUE.

  Jeśli ***zmniejsz*** maksymalną długość komunikatu w istniejącej kolejce, to istniejące komunikaty nie będą miały wpływu na tę wartość. Wszystkie nowe komunikaty muszą jednak spełniać nowe kryteria.

Czyszczenie kolejki lokalnej

Aby wyczyścić kolejkę lokalną, można użyć komendy CLEAR.

Aby usunąć wszystkie komunikaty z kolejki lokalnej o nazwie MAGENTA.QUEUE, użyj następującej komendy:

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

Uwaga: Nie ma podpowiedzi, która pozwala Ci zmienić zdanie; po naciśnięciu klawisza Enter wiadomości zostaną utracone.

Nie można wyczyścić kolejki, jeśli:

- Istnieją niezatwierdzone komunikaty, które zostały umieszczone w kolejce w punkcie synchronizacji.
- Aplikacja aktualnie ma otwartą kolejkę.

Usuwanie kolejki lokalnej

Aby usunąć kolejkę lokalną, można użyć komendy MQSC DELETE QLOCAL.

Nie można usunąć kolejki, jeśli na niej nie ma zatwierdzonych komunikatów. Jeśli jednak kolejka ma jeden lub więcej zatwierdzonych komunikatów i nie ma żadnych niezatwierdzonych komunikatów, może zostać usunięta tylko wtedy, gdy zostanie podana opcja PURGE. Na przykład:

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

Podanie wartości NOPURGE zamiast PURGE zapewnia, że kolejka nie zostanie usunięta, jeśli zawiera jakiekolwiek zatwierdzone komunikaty.

Przeglądanie kolejek

Produkt WebSphere MQ udostępnia przykładową przeglądarkę kolejek, której można używać do przeglądania zawartości komunikatów w kolejce. Przeglądarka jest dostarczana zarówno w formacie źródłowym, jak i w formacie wykonywalnym.

MQ_INSTALLATION_PATH reprezentuje katalog najwyższego poziomu, w którym zainstalowany jest produkt WebSphere MQ .

W produkcie WebSphere MQ for Windowsnazwy plików i ścieżki dla przeglądarki kolejki przykładowej są następujące:

Źródło
MQ_INSTALLATION_PATH\tools\c\samples\

Wykonywalny
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

W produkcie WebSphere MQ for UNIX and Linuxnazwy plików i ścieżki są następujące:

Źródło
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

Wykonywalny
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

Przykład wymaga dwóch parametrów wejściowych, nazwy kolejki i nazwy menedżera kolejek. Na przykład:

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

Typowe wyniki tej komendy są przedstawione w sekcji Rysunek 16 na stronie 90.

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 850
  Format : 'MQEVENT '
  Priority : 0  Persistence : 0
  MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                        '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '104'

****   Message      ****

 length - 104 bytes

00000000:  0700 0000 2400 0000 0100 0000 2C00 0000 '....→.......,...'
00000010:  0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020:  0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030:  0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040:  7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager    '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020                     '                '

 No more messages
 MQCLOSE
 MQDISC
```

Rysunek 16. Typowe wyniki w przeglądarce kolejek

Włączanie dużych kolejek

IBM WebSphere MQ obsługuje kolejki większe niż 2 GB.

W systemach Windows obsługa dużych plików jest dostępna bez dodatkowego włączenia. W systemach AIX, HP-UX, Linuxi Solaris konieczne jest jawne włączenie obsługi dużych plików, zanim możliwe będzie utworzenie plików kolejki większych niż 2 GB. Informacje na temat sposobu wykonania tej czynności można znaleźć w dokumentacji systemu operacyjnego.

Niektóre programy narzędziowe, takie jak tar, nie mogą sobie poradzić z plikami większymi niż 2 GB. Przed włączeniem obsługi dużych plików należy zapoznać się z dokumentacją systemu operacyjnego, aby uzyskać informacje na temat ograniczeń w użyciu programów narzędziowych.

Informacje na temat planowania ilości pamięci masowej dla kolejek można znaleźć w serwisie WWW produktu IBM WebSphere MQ pod kątem raportów dotyczących wydajności poszczególnych platform:

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

Praca z kolejkami aliasowymi

Kolejkę aliasową można zdefiniować tak, aby odwoływał się pośrednio do innej kolejki lub tematu.

V 7.5.0.8

Ostrzeżenie: Listy dystrybucyjne nie obsługują korzystania z kolejek aliasowych, które wskazują na obiekty tematów. Począwszy od wersji Version 7.5.0, Fix Pack 8, jeśli kolejka aliasowa wskazuje na obiekt tematu na liście dystrybucyjnej, produkt IBM WebSphere MQ zwraca błąd MQRC_ALIAS_BASE_Q_TYPE_ERROR.

Kolejka, do której odwołuje się kolejka aliasowa, może być jedną z następujących kolejek:

- Kolejka lokalna (patrz sekcja “Definiowanie kolejki lokalnej” na stronie 86).
- Lokalna definicja kolejki zdalnej (patrz sekcja “Tworzenie lokalnej definicji kolejki zdalnej” na stronie 116).
- Temat.

Kolejka aliasowa nie jest kolejką rzeczywistą, ale definicją, która jest tłumaczona na kolejkę rzeczywistą (lub docelową) w czasie wykonywania. Definicja kolejki aliasowej określa kolejkę docelową. Gdy aplikacja wykonuje wywołanie programu MQOPEN do kolejki aliasowej, menedżer kolejek tłumaczy alias na nazwę kolejki docelowej.

Kolejka aliasowa nie może zostać przetłumaczona na inną lokalnie zdefiniowaną kolejkę aliasową. Kolejka aliasowa może jednak zostać przetłumaczona na kolejki aliasowe zdefiniowane w innym miejscu w klastrach, do których należy menedżer kolejek lokalnych. Więcej informacji na ten temat zawiera sekcja Rozstrzyganie nazw .

Kolejki aliasowe są przydatne w następujących sytuacjach:

- Nadawanie różnym aplikacjom różnych poziomów uprawnień dostępu do kolejki docelowej.
- Zezwalanie różnym aplikacjom na pracę z tą samą kolejką na różne sposoby. (Być może chcesz przypisać różne domyślne priorytety lub różne domyślne wartości trwałości).
- Upraszcza konserwację, migrację i równoważenie obciążenia. (Być może chcesz zmienić nazwę kolejki docelowej bez konieczności zmiany aplikacji, która nadal używa aliasu).

Na przykład załóżmy, że została opracowana aplikacja do umieszczania komunikatów w kolejce o nazwie MY.ALIAS.QUEUE. Określa ona nazwę tej kolejki podczas wysyłania żądania MQOPEN i pośrednio, jeśli umieszcza ona komunikat w tej kolejce. Aplikacja nie wie, że kolejka jest kolejką aliasową. Dla każdego wywołania MQI używającego tego aliasu menedżer kolejek tłumaczy rzeczywistą nazwę kolejki, która może być kolejką lokalną lub zdalną zdefiniowaną w tym menedżerze kolejek.

Zmieniając wartość atrybutu TARGET, można przekierować wywołania MQI do innej kolejki, prawdopodobnie w innym menedżerze kolejek. Jest to przydatne w przypadku konserwacji, migracji i równoważenia obciążenia.

Definiowanie kolejki aliasowej

Następująca komenda tworzy kolejkę aliasową:

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

Ta komenda przekierowuje wywołania MQI, które określają parametr MY.ALIAS.QUEUE do kolejki YELLOW.QUEUE. Komenda nie tworzy kolejki docelowej; wywołania MQI nie powiodą się, jeśli kolejka YELLOW.QUEUE nie istnieje w czasie wykonywania.

W przypadku zmiany definicji aliasu można przekierować wywołania MQI do innej kolejki. Na przykład:

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

Ta komenda przekierowuje wywołania MQI do innej kolejki MAGENTA.QUEUE.

Można również użyć kolejek aliasowych, aby pojedyncza kolejka (kolejka docelowa) miała różne atrybuty dla różnych aplikacji. W tym celu należy zdefiniować dwa aliasy, po jednym dla każdej aplikacji. Załóżmy, że istnieją dwie aplikacje:

- Aplikacja ALPHA może umieszczać komunikaty w kolorze YELLOW.QUEUE, ale nie jest dozwolone pobieranie z niego komunikatów.
- Aplikacja BETA może pobrać komunikaty z YELLOW.QUEUE, ale nie jest dozwolone umieszczanie w nim komunikatów.

Poniższa komenda definiuje alias, który jest włączony i wyłączony dla aplikacji ALPHA:

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

Poniższa komenda definiuje alias, który jest wyłączany i włączany dla BETA aplikacji:

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

ALPHA używa nazwy kolejki ALPHAS.ALIAS.QUEUE w wywołaniach MQI; BETA używa nazwy kolejki BETAS.ALIAS.QUEUE. Obie mają dostęp do tej samej kolejki, ale na różne sposoby.

Podczas definiowania aliasów kolejek można używać atrybutów LIKE i REPLACE w taki sam sposób, jak w przypadku kolejek lokalnych.

Używanie innych komend z kolejkami aliasowymi

Aby wyświetlić lub zmienić atrybuty kolejki aliasowej lub usunąć obiekt kolejki aliasowej, można użyć odpowiednich komend MQSC. Na przykład:

Użyj następującej komendy, aby wyświetlić atrybuty kolejki aliasowej:

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

Użyj następującej komendy, aby zmienić nazwę kolejki podstawowej, na którą jest tłumaczony alias, gdzie opcja `force` wymusza zmianę, nawet jeśli kolejka jest otwarta:

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

Użyj następującej komendy, aby usunąć ten alias kolejki:

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

Nie można usunąć kolejki aliasowej, jeśli aplikacja ma obecnie otwartą kolejkę. Więcej informacji na temat tej i innych komend kolejki aliasowej zawiera dokumentacja Informacje dodatkowe dotyczące komend MQSC .

Praca z kolejkami modelami

Menedżer kolejek tworzy *kolejkę dynamiczną* , jeśli odbierze wywołanie MQI z aplikacji, określając nazwę kolejki, która została zdefiniowana jako kolejka modelowa. Nazwa nowej kolejki dynamicznej jest generowana przez menedżer kolejek po utworzeniu kolejki. *Kolejka modelowa* jest to szablon, który określa atrybuty wszystkich kolejek dynamicznych utworzonych na podstawie tego szablonu. Kolejki modelowe stanowią wygodną metodę tworzenia kolejek zgodnie z wymaganiami.

Definiowanie kolejki modelowej

Kolejkę modelową można zdefiniować za pomocą zestawu atrybutów w taki sam sposób, w jaki definiuje się kolejkę lokalną. Kolejki modelowe i kolejki lokalne mają ten sam zestaw atrybutów, z tym wyjątkiem, że w kolejkach modelowych można określić, czy utworzone kolejki dynamiczne mają charakter tymczasowy, czy trwały. (Kolejki stałe są obsługiwane przez restarty menedżera kolejek, tymczasowe nie są dostępne). Na przykład:

```
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)
```

Ta komenda tworzy definicję kolejki modelowej. Z atrybutu DEFTYPE można zauważyć, że rzeczywiste kolejki utworzone na podstawie tego szablonu są trwałymi kolejkami dynamicznymi. Wszystkie nieokreślone atrybuty są automatycznie kopiowane ze strony SYSYTEM.DEFAULT.MODEL.QUEUE , kolejka domyślna.

Atrybuty LIKE i REPLACE mogą być używane podczas definiowania kolejek modelowych, w taki sam sposób, w jaki są używane z kolejkami lokalnymi.

Korzystanie z innych komend z kolejkami modelami

Za pomocą odpowiednich komend MQSC można wyświetlać lub zmieniać atrybuty kolejki modelowej, a także usuwać obiekty kolejki modelowej. Na przykład:

Aby wyświetlić atrybuty kolejki modelowej, należy użyć następującej komendy:

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

Użyj następującej komendy, aby zmienić model w celu włączenia operacji umieszczania w dowolnej kolejce dynamicznej utworzonej na podstawie tego modelu:

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

Aby usunąć tę kolejkę modelową, użyj następującej komendy:

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

Praca z tematami administracyjnymi

Aby zarządzać tematami administracyjnymi, należy użyć komend MQSC.

Szczegółowe informacje na temat tych komend można znaleźć w sekcji Informacje dodatkowe dotyczące komend MQSC .

Pojęcia pokrewne

Administracyjne obiekty tematów

“Definiowanie tematu administracyjnego” na stronie 94
Aby utworzyć temat administracyjny, należy użyć komendy MQSC **DEFINE TOPIC** . Definiując temat administracyjny, można opcjonalnie ustawić każdy atrybut tematu.

“Wyświetlanie atrybutów obiektu tematu administracyjnego” na stronie 94
Aby wyświetlić obiekt tematu administracyjnego, należy użyć komendy MQSC **DISPLAY TOPIC** .

“Zmiana atrybutów tematu administracyjnego” na stronie 95
Atrybuty tematu można zmieniać na dwa sposoby, korzystając z komendy **ALTER TOPIC** lub **DEFINE TOPIC** z atrybutem **REPLACE** .

“Kopiowanie definicji tematu administracyjnego” na stronie 95
Definicję tematu można skopiować za pomocą atrybutu LIKE w komendzie **DEFINE** .

“Usuwanie definicji tematu administracyjnego” na stronie 96
Aby usunąć temat administracyjny, można użyć komendy MQSC **DELETE TOPIC** .

Definiowanie tematu administracyjnego

Aby utworzyć temat administracyjny, należy użyć komendy MQSC **DEFINE TOPIC** . Definiując temat administracyjny, można opcjonalnie ustawić każdy atrybut tematu.

Dowolny atrybut tematu, który nie jest jawnie ustawiony, jest dziedziczony z domyślnego tematu administracyjnego, SYSTEM.DEFAULT.TOPIC(temat), który został utworzony podczas instalacji systemu.

Na przykład komenda **DEFINE TOPIC** , która jest następująca, definiuje temat o nazwie **ORANGE.TOPIC** o następujących parametrach:

- Rozwiązuje się w łańcuchu tematu ORANGE. Więcej informacji na temat sposobu użycia łańcuchów tematów zawiera sekcja Łączenie łańcuchów tematów.
- Każdy atrybut, który jest ustawiony na ASPARENT, używa atrybutu zdefiniowanego przez nadrzędny temat w tym temacie. To działanie jest powtarzane w drzewie tematów, jeśli jest to główny temat, SYSTEM.BASE.TOPIC . Więcej informacji na temat drzew tematów zawiera sekcja Drzewa tematów.

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

Uwaga:

- Wszystkie wartości atrybutów są wartościami domyślnymi, z wyjątkiem wartości w łańcuchu tematu. Są one pokazywane tutaj tylko jako ilustracja. Można je pominąć, jeśli użytkownik ma pewność, że wartości domyślne są takie, które mają zostać zmienione lub które nie zostały zmienione. Patrz także “Wyświetlanie atrybutów obiektu tematu administracyjnego” na stronie 94.
- Jeśli w tym samym menedżerze kolejek znajduje się już temat administracyjny o nazwie ORANGE.TOPIC-ta komenda nie powiedzie się. Użyj atrybutu REPLACE, jeśli chcesz nadpisać istniejącą definicję tematu, ale patrz także “Zmiana atrybutów tematu administracyjnego” na stronie 95

Wyświetlanie atrybutów obiektu tematu administracyjnego

Aby wyświetlić obiekt tematu administracyjnego, należy użyć komendy MQSC **DISPLAY TOPIC** .

Aby wyświetlić wszystkie tematy, należy użyć:

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

Można selektywnie wyświetlać atrybuty, określając je pojedynczo.Na przykład:

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

```
        DEFPRTY +
        NPMSGDLV
```

Ta komenda wyświetla trzy określone atrybuty w następujący sposób:

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

Aby wyświetlić temat wartości ASPARENT, ponieważ są one używane w czasie wykonywania, należy użyć komendy DISPLAY TPSTATUS. Na przykład:

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

Komenda wyświetla następujące szczegóły:

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

Po zdefiniowaniu tematu administracyjnego wszystkie atrybuty, które nie zostały jawnie określone w domyślnym temacie administracyjnym, są określane jako SYSTEM.DEFAULT.TOPIC. Aby sprawdzić, jakie są te atrybuty domyślne, należy użyć następującej komendy:

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

Zmiana atrybutów tematu administracyjnego

Atrybuty tematu można zmieniać na dwa sposoby, korzystając z komendy **ALTER TOPIC** lub **DEFINE TOPIC** z atrybutem **REPLACE** .

Jeśli na przykład ma zostać zmieniony domyślny priorytet komunikatów dostarczanych do tematu o nazwie ORANGE.TOPIC, aby mieć wartość 5, należy użyć jednej z następujących komend.

- Za pomocą komendy **ALTER** :

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

  Ta komenda zmienia pojedynczy atrybut, który ma domyślny priorytet komunikatu dostarczanego do tego tematu na wartość 5; wszystkie pozostałe atrybuty pozostają takie same.
- Za pomocą komendy **DEFINE** :

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

  Ta komenda zmienia domyślny priorytet komunikatów dostarczonych do tego tematu. Wszystkie pozostałe atrybuty mają podane wartości domyślne.

Jeśli priorytet komunikatów wysyłanych do tego tematu zostanie zmieniony, nie ma to wpływu na istniejące komunikaty. Jednak każdy nowy komunikat używa podanego priorytetu, jeśli nie jest on udostępniany przez aplikację publikowania.

Kopiowanie definicji tematu administracyjnego

Definicję tematu można skopiować za pomocą atrybutu LIKE w komendzie **DEFINE** .

Na przykład:

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

Ta komenda powoduje utworzenie tematu MAGENTA.TOPIC, z tymi samymi atrybutami, co w oryginalnym temacie, ORANGE.TOPIC, a nie te z domyślnego tematu administracyjnego systemu. Wprowadź nazwę

tematu, który ma zostać skopiowany dokładnie tak, jak został wprowadzony podczas tworzenia tematu. Jeśli nazwa zawiera małe litery, należy ująć ją w znaki pojedynczego cudzysłowu.

Można również użyć tego formularza komendy **DEFINE** , aby skopiować definicję tematu, ale dokonać zmian w atrybutach oryginału. Na przykład:

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

Istnieje również możliwość skopiowania atrybutów tematu BLUE.TOPIC (TEMAT) do tematu GREEN.TOPIC (temat) i określa, że gdy publikacje nie mogą być dostarczane do odpowiedniej kolejki subskrybenta, nie są umieszczane w kolejce niedostarczonych komunikatów. Na przykład:

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

Usuwanie definicji tematu administracyjnego

Aby usunąć temat administracyjny, można użyć komendy MQSC **DELETE TOPIC** .

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

Aplikacje nie będą mogły otworzyć tematu w celu opublikowania lub udostępnienia nowych subskrypcji przy użyciu nazwy obiektu ORANGE.TOPIC. Publikowanie aplikacji, które mają otwarty temat, są w stanie kontynuować publikowanie przetłumaczanego łańcucha tematu. Wszystkie subskrypcje już wprowadzone w tym temacie będą nadal otrzymywać publikacje po usunięciu tematu.

Aplikacje, które nie odwołują się do tego obiektu tematu, ale używają rozstrzygniętego łańcucha tematu, który ten obiekt tematu reprezentował, 'ORANGE' w tym przykładzie, kontynuują pracę. W tym przypadku dziedziczą właściwości z obiektu tematu wyżej w drzewie tematów. Więcej informacji na temat drzew tematów zawiera sekcja Drzewa tematów.

Praca z subskrypcjami

Do zarządzania subskrypcjami można używać komend MQSC.

Subskrypcje mogą być jednym z trzech typów zdefiniowanych w atrybucie **SUBTYPE** :

ADMINISTRATOR
Administracyjnie definiowane przez użytkownika.

PROXY
Subskrypcja utworzona wewnętrznie w celu kierowania publikacji między menedżerami kolejek.

Interfejs API
Tworzone programowo, na przykład przy użyciu wywołania MQSUB MQI.

Szczegółowe informacje na temat tych komend można znaleźć w publikacji Informacje dodatkowe dotyczące komend MQSC .

Pojęcia pokrewne

“Definiowanie subskrypcji administracyjnej” na stronie 97
Aby utworzyć subskrypcję administracyjną, należy użyć komendy MQSC **DEFINE SUB** . Można również użyć wartości domyślnej zdefiniowanej w definicji domyślnej subskrypcji lokalnej. Można także zmodyfikować parametry subskrypcji z poziomu domyślnej subskrypcji lokalnej SYSTEM.DEFAULT.SUB , który został utworzony podczas instalowania systemu.

“Wyświetlanie atrybutów subskrypcji” na stronie 97
Za pomocą komendy **DISPLAY SUB** można wyświetlić skonfigurowane atrybuty dowolnej subskrypcji znanej menedżerowi kolejek.

“Zmiana atrybutów subskrypcji lokalnej” na stronie 98

Atrybuty subskrypcji można zmienić na dwa sposoby, korzystając z komendy **ALTER SUB** lub **DEFINE SUB** z atrybutem **REPLACE** .

“Kopiowanie definicji subskrypcji lokalnej” na stronie 99
Definicję subskrypcji można skopiować za pomocą atrybutu **LIKE** w komendzie **DEFINE** .

“Usuwanie subskrypcji” na stronie 99
Aby usunąć subskrypcję lokalną, można użyć komendy MQSC **DELETE SUB** .

Definiowanie subskrypcji administracyjnej

Aby utworzyć subskrypcję administracyjną, należy użyć komendy MQSC **DEFINE SUB** . Można również użyć wartości domyślnej zdefiniowanej w definicji domyślnej subskrypcji lokalnej. Można także zmodyfikować parametry subskrypcji z poziomu domyślnej subskrypcji lokalnej SYSTEM.DEFAULT.SUB , który został utworzony podczas instalowania systemu.

Na przykład następująca komenda **DEFINE SUB** definiuje subskrypcję o nazwie ORANGE o następujących parametrach:

- Trwała subskrypcja, co oznacza, że utrzymuje się ona po restarcie menedżera kolejek z nieograniczonym terminem utraty ważności.
- Otrzymuj publikacje wykonane w łańcuchu tematu ORANGE z priorytetami komunikatów ustawionym przez aplikacje publikowania.
- Publikacje dostarczone dla tej subskrypcji są wysyłane do kolejki lokalnej SUBQ, a ta kolejka musi być zdefiniowana przed zdefiniowaniu subskrypcji.

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

Uwaga:

- Subskrypcja i nazwa łańcucha tematu nie muszą być zgodne.
- Wszystkie wartości atrybutów są wartościami domyślnymi, z wyjątkiem wartości opisu i łańcucha tematu. Są one pokazywane tutaj tylko jako ilustracja. Można je pominąć, jeśli użytkownik ma pewność, że wartości domyślne są takie, które mają zostać zmienione lub które nie zostały zmienione. Patrz także “Wyświetlanie atrybutów subskrypcji” na stronie 97.
- Jeśli w tym samym menedżerze kolejek istnieje już subskrypcja lokalna o nazwie TEST, wykonanie tej komendy nie powiedzie się. Użyj atrybutu **REPLACE** , jeśli chcesz nadpisać istniejącą definicję kolejki, ale zobacz także “Zmiana atrybutów subskrypcji lokalnej” na stronie 98.
- Jeśli kolejka SUBQ nie istnieje, wykonanie tej komendy nie powiedzie się.

Wyświetlanie atrybutów subskrypcji

Za pomocą komendy **DISPLAY SUB** można wyświetlić skonfigurowane atrybuty dowolnej subskrypcji znanej menedżerowi kolejek.

Na przykład:

```
DISPLAY SUB (ORANGE)
```

Można selektywnie wyświetlać atrybuty, określając je pojedynczo. Na przykład:

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

Ta komenda wyświetla trzy określone atrybuty w następujący sposób:

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

TOPICSTR jest rozstrzygniętym łańcuchem tematu, na którym działa ten subskrybent. Gdy subskrypcja jest zdefiniowana w celu użycia obiektu tematu, łańcuch tematu z tego obiektu jest używany jako przedrostek do łańcucha tematu udostępnianego podczas wykonywania subskrypcji. Identyfikator SUBID jest unikalnym identyfikatorem przypisanym przez menedżer kolejek po utworzeniu subskrypcji. Jest to przydatny atrybut do wyświetlenia, ponieważ niektóre nazwy subskrypcji mogą być długie lub w różnych zestawach znaków, dla których może stać się niepraktyczne.

Alternatywną metodą wyświetlania subskrypcji jest użycie identyfikatora SUBID:

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

Ta komenda daje te same dane wyjściowe, co wcześniej:

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

Subskrypcje proxy w menedżerze kolejek nie są domyślnie wyświetlane. Aby wyświetlić je, należy określić **SUBTYPE** serwera PROXY lub ALL.

Aby wyświetlić atrybuty środowiska wykonawczego, można użyć komendy DISPLAY SBSTATUS . Na przykład użyj komendy:

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

Zostaną wyświetlone następujące dane wyjściowe:

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

Po zdefiniowaniu subskrypcji administracyjnej wszystkie atrybuty, które nie zostały jawnie określone z subskrypcji domyślnej, są określane jako SYSTEM.DEFAULT.SUB. Aby sprawdzić, jakie są te atrybuty domyślne, należy użyć następującej komendy:

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

Zmiana atrybutów subskrypcji lokalnej

Atrybuty subskrypcji można zmienić na dwa sposoby, korzystając z komendy **ALTER SUB** lub **DEFINE SUB** z atrybutem **REPLACE** .

Jeśli na przykład chcesz zmienić priorytet komunikatów dostarczanych do subskrypcji o nazwie ORANGE na 5, użyj jednej z następujących komend:

- Za pomocą komendy ALTER:

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

  Ta komenda zmienia pojedynczy atrybut, który ma priorytet komunikatów dostarczanych do tej subskrypcji na 5; wszystkie pozostałe atrybuty pozostają takie same.

- Za pomocą komendy DEFINE:

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

Ta komenda zmienia nie tylko priorytet komunikatów dostarczanych do tej subskrypcji, ale wszystkie inne atrybuty, które mają podane wartości domyślne.

Jeśli użytkownik zmieni priorytet komunikatów wysyłanych do tej subskrypcji, nie ma to wpływu na istniejące komunikaty. Wszystkie nowe komunikaty mają jednak określony priorytet.

Kopiowanie definicji subskrypcji lokalnej

Definicję subskrypcji można skopiować za pomocą atrybutu **LIKE** w komendzie **DEFINE** .

Na przykład:

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

Można również skopiować atrybuty elementu podrzędnego REAL do podkomendy THIRD.SUBi określić, że identyfikatorem correlID dostarczonym publikacji jest THIRD, a nie publikatorów correlID. Na przykład:

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

Usuwanie subskrypcji

Aby usunąć subskrypcję lokalną, można użyć komendy MQSC **DELETE SUB** .

```
DELETE SUB(ORANGE)
```

Subskrypcję można również usunąć za pomocą identyfikatora SUBID:

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

Sprawdzanie komunikatów w subskrypcji

O tym zadaniu

Gdy zdefiniowana jest subskrypcja, jest ona powiązana z kolejką. Opublikowane komunikaty zgodne z tą subskrypcją są umieszczane w tej kolejce.

Należy pamiętać, że następujące komendy **runmqsc** pokazują tylko te subskrypcje, które otrzymały komunikaty.

Aby sprawdzić komunikaty znajdujące się obecnie w kolejce w celu wykonania subskrypcji, wykonaj następujące kroki:

Procedura

1. Informacje na temat sprawdzania komunikatów w kolejce dla typu subskrypcji **DISPLAY SBSTATUS(<sub_name>) NUMMSGS**zawiera sekcja “Wyświetlanie atrybutów subskrypcji” na stronie 97.
2. Jeśli wartość **NUMMSGS** jest większa od zera, należy zidentyfikować kolejkę powiązaną z subskrypcją, wpisując **DISPLAY SUB(<sub_name>)DEST**.
3. Za pomocą nazwy kolejki zwróconej można wyświetlić komunikaty zgodnie z techniką opisaną w sekcji “Przeglądanie kolejek” na stronie 89.

Praca z usługami

Obiekty usług są środkiem, za pomocą którego można zarządzać dodatkowymi procesami jako część menedżera kolejek. Za pomocą usług można definiować programy, które są uruchamiane i zatrzymane

po uruchomieniu i zakończeniu menedżera kolejek. Usługi produktu IBM WebSphere MQ są zawsze uruchamiane przy użyciu identyfikatora użytkownika, który uruchomił menedżer kolejek.

Obiekty usług mogą być jednym z następujących typów:

Serwer
Serwer jest obiektem usługi, który ma parametr SERVTYPE określony jako SERVER. Obiekt usługi serwera jest to definicja programu, który jest wykonywany podczas uruchamiania określonego menedżera kolejek. Obiekty usług serwera definiują programy, które zwykle są uruchamiane przez długi czas. Na przykład obiekt usługi serwera może być używany do wykonywania procesu monitora wyzwalacza, takiego jak `runmqtrm`.

Jednocześnie może być uruchamiana tylko jedna instancja obiektu usługi serwera. Status uruchomionych obiektów usług serwera może być monitorowany za pomocą komendy MQSC DISPLAY SVSTATUS.

Komenda
Komenda jest obiektem usługi, który ma parametr SERVTYPE określony jako COMMAND. Obiekty usług komend są podobne do obiektów usług serwera, jednak wiele instancji obiektu usługi komendy może być uruchamiane współbieżnie, a ich status nie może być monitorowany za pomocą komendy MQSC DISPLAY SVSTATUS.

Jeśli komenda MQSC ZATRZYMAJ USŁUGĘ nie jest wykonywana, aby określić, czy program uruchomiony przez komendę MQSC, START SERVICE, jest nadal aktywny, przed wykonaniem programu zatrzymania.

Definiowanie obiektu usługi

Obiekt usługi jest definiowany przy użyciu różnych atrybutów.

Atrybuty są następujące:

SERVTYPE
Definiuje typ obiektu usługi. Lista poprawnych wartości:

SERVER
Obiekt usługi serwera.

W danym momencie może być wykonywana tylko jedna instancja obiektu usługi serwera. Status obiektów usług serwera może być monitorowany za pomocą komendy MQSC DISPLAY SVSTATUS.

KOMENDA
Obiekt usługi komendy.

Wiele instancji obiektu usługi komendy może być wykonywanych współbieżnie. Status obiektów usługi komendy nie może być monitorowany.

STARTCMD
Program, który jest wykonywany w celu uruchomienia usługi. Należy podać pełną ścieżkę do programu.

STARTARG
Argumenty przekazane do programu startowego.

STDERR
Określa ścieżkę do pliku, do którego ma zostać przekierowany błąd standardowy (stderr) programu usługowego.

STDOUT
Określa ścieżkę do pliku, do którego powinny zostać przekierowane standardowe wyjście (wyjście standardowe) programu usługowego.

STOPCMD
Program, który jest wykonywany w celu zatrzymania usługi. Należy podać pełną ścieżkę do programu.

STOPARG
Argumenty przekazane do programu zatrzymania.

CONTROL
Określa sposób, w jaki usługa ma być uruchomiona i zatrzymana:

RĘCZNE
Usługa nie jest automatycznie uruchamiana lub zatrzymana automatycznie. Sterowanie jest kontrolowane za pomocą komend START SERVICE i STOP SERVICE. Jest to wartość domyślna.

QMGR
Zdefiniowana usługa jest uruchamiana i zatrzymana w tym samym czasie, co menedżer kolejek, który jest uruchamiany i zatrzymany.

TYLKO startonly
Usługa ma zostać uruchomiona w tym samym czasie co menedżer kolejek, ale nie jest wymagana do zatrzymania, gdy menedżer kolejek jest zatrzymany.

Pojęcia pokrewne

“Zarządzanie usługami” na stronie 101
Użycie parametru CONTROL powoduje, że instancja obiektu usługi może być uruchamiana i zatrzymana automatycznie przez menedżer kolejek lub uruchomiona i zatrzymana za pomocą komend MQSC START SERVICE i STOP SERVICE.

Zarządzanie usługami

Użycie parametru CONTROL powoduje, że instancja obiektu usługi może być uruchamiana i zatrzymana automatycznie przez menedżer kolejek lub uruchomiona i zatrzymana za pomocą komend MQSC START SERVICE i STOP SERVICE.

Gdy instancja obiektu usługi jest uruchamiana, w dzienniku błędów menedżera kolejek zapisywany jest komunikat zawierający nazwę obiektu usługi i identyfikator procesu uruchomionego procesu. Poniżej przedstawiono przykładową pozycję dziennika dla obiektu usługi serwera:

```
  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.
```

Przykładowy wpis dziennika dla obiektu usługi komendy jest następujący:

```
  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.
```

Gdy usługa serwera instancji zostanie zatrzymana, do dzienników błędów menedżera kolejek zostanie zapisany komunikat zawierający nazwę usługi i identyfikator procesu zakończenia. Poniżej znajduje się przykładowa pozycja dziennika dla obiektu usługi serwera:

```
  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.
```

Odsyłacze pokrewne

“Dodatkowe zmienne środowiskowe” na stronie 102
Po uruchomieniu usługi środowisko, w którym uruchamiany jest proces usługi, jest dziedziczone ze środowiska menedżera kolejek. Istnieje możliwość zdefiniowania dodatkowych zmiennych środowiskowych, które mają zostać ustawione w środowisku procesu usługi, przez dodanie zmiennych, które mają zostać zdefiniowane, do jednego z plików przesłaniania środowiska `service.env` .

Dodatkowe zmienne środowiskowe

Po uruchomieniu usługi środowisko, w którym uruchamiany jest proces usługi, jest dziedziczone ze środowiska menedżera kolejek. Istnieje możliwość zdefiniowania dodatkowych zmiennych środowiskowych, które mają zostać ustawione w środowisku procesu usługi, przez dodanie zmiennych, które mają zostać zdefiniowane, do jednego z plików przesłaniania środowiska `service.env` .

Uwaga:

Istnieją dwa możliwe pliki, do których można dodać zmienne środowiskowe:

- Plik `service.env` o zasięgu komputera, który znajduje się w katalogu `/var/mqm` w systemach UNIX and Linux lub w katalogu danych wybranym podczas instalacji w systemach Windows .
- Plik zasięgu menedżera kolejek `service.env` , który znajduje się w katalogu danych menedżera kolejek. Na przykład położenie pliku przesłaniania środowiska dla menedżera kolejek o nazwie `QMNAME` jest następujące:
  - W systemach UNIX and Linux : `/var/mqm/qmgrs/QMNAME/service.env`
  - W systemach Windows : `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`

Oba pliki są przetwarzane, jeśli są dostępne, z definicjami w pliku zasięgu menedżera kolejek, które mają pierwszeństwo przed definicjami w pliku zasięgu komputera.

W pliku `service.env`można podać dowolną zmienną środowiskową. Jeśli na przykład usługa systemu IBM WebSphere MQ uruchamia kilka komend, przydatne może być ustawienie zmiennej użytkownika PATH w pliku `service.env` . Wartości ustawione dla zmiennej nie mogą być zmiennymi środowiskowymi; na przykład CLASSPATH=*%CLASSPATH%* jest niepoprawna. Podobnie w przypadku ścieżki Linux PATH=*$PATH*`:/opt/mqm/bin` dałoby nieoczekiwane wyniki.

Zmienna CLASSPATH musi być zapisana wielkimi literami, a instrukcja ścieżki klasy może zawierać tylko literały. Niektóre usługi (na przykład Telemetria) ustawiają własną ścieżkę klasy. Zmienna CLASSPATH zdefiniowana w zmiennej `service.env` zostanie do niej dodana.

Format zmiennych zdefiniowanych w pliku`service.env` jest listą par nazw i wartości zmiennych. Każda zmienna musi być zdefiniowana w nowym wierszu, a każda zmienna jest przyjmowana tak, jak jest jawnie zdefiniowana, łącznie z białymi spacjami. Przykład pliku`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
```

Odsyłacze pokrewne

“Wymienne wstawki w definicjach usług” na stronie 103
W definicji obiektu usługi możliwe jest podstawienia leksemów. Znaczniki, które są zastępowane, są automatycznie zastępowane rozszerzonym tekstem, gdy wykonywany jest program usługowy. Znaczniki zastępcze mogą być pobierane z następującej listy wspólnych znaczników lub ze wszystkich zmiennych zdefiniowanych w pliku service.env.

Wymienne wstawki w definicjach usług

W definicji obiektu usługi możliwe jest podstawienia leksemów. Znaczniki, które są zastępowane, są automatycznie zastępowane rozszerzonym tekstem, gdy wykonywany jest program usługowy. Znaczniki zastępcze mogą być pobierane z następującej listy wspólnych znaczników lub ze wszystkich zmiennych zdefiniowanych w pliku service.env.

Poniżej znajdują się wspólne znaczniki, których można użyć do podstawienia znaczników w definicji obiektu usługi:

ŚCIEŻKA_INSTALACJI_MQ
Lokalizacja, w której zainstalowany jest produkt WebSphere MQ .

ŚCIEŻKA_DATOWANA_MQ
Położenie katalogu danych produktu WebSphere MQ :

- W systemach UNIX and Linux położeniem katalogu danych produktu WebSphere MQ jest `/var/mqm/`
- W systemach Windows położenie katalogu danych produktu WebSphere MQ jest katalogiem danych wybranym podczas instalacji produktu WebSphere MQ .

QMNAME
Nazwa bieżącego menedżera kolejek.

MQ_SERVICE_NAME
Nazwa danej usługi.

PID MQ_SERVER_PID
Ten znacznik może być używany tylko przez argumenty STOPARG i STOPCMD.

W przypadku obiektów usług serwera ten znacznik jest zastępowany identyfikatorem procesu uruchamianego przez argumenty STARTCMD i STARTARG. W przeciwnym razie znacznik ten jest zastępowany wartością 0.

MQ_Q_MGR_DATA_PATH
Położenie katalogu danych menedżera kolejek.

MQ_Q_MGR_DATA_NAME
Przekształcona nazwa menedżera kolejek. Więcej informacji na temat transformacji nazw zawiera sekcja Informacje o nazwach plików WebSphere MQ.

Aby użyć wymiennych wstawień, należy wstawić znacznik w obrębie + znaków do dowolnego łańcucha STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT lub STDERR. Przykłady tego programu znajdują się w sekcji “Przykłady korzystania z obiektów usług” na stronie 103.

Przykłady korzystania z obiektów usług

Usługi w tej sekcji są zapisywane z separatorami ścieżek w stylu systemu UNIX , z wyjątkiem sytuacji, w których określono inaczej.

Korzystanie z obiektu usługi serwera

W tym przykładzie przedstawiono sposób definiowania, używania i zmieniania obiektu usługi serwera w celu uruchomienia monitora wyzwalacza.

1. Obiekt usługi serwera jest zdefiniowany za pomocą następującej komendy MQSC:

```
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+')
```

gdzie:

+MQ_INSTALL_PATH+ jest tokenem reprezentującym katalog instalacyjny.

+QMNAME+ jest znacznikiem reprezentującym nazwę menedżera kolejek.

ACCOUNTS.INITIATION.QUEUE jest kolejką inicjują.

Program amqsstop jest przykładowym programem dostarczonym razem z produktem WebSphere MQ , który żąda od menedżera kolejek przerwania wszystkich połączeń dla identyfikatora procesu. Program amqsstop generuje komendy PCF, dlatego serwer komend musi być uruchomiony.

+MQ_SERVER_PID+ jest znacznikiem reprezentującym identyfikator procesu przekazany do programu zatrzymania.

Listę wspólnych znaczników można znaleźć w sekcji “Wymienne wstawki w definicjach usług” na stronie 103 .

2. Instancja obiektu usługi serwera zostanie wykonana po następnym uruchomieniu menedżera kolejek. Jednak natychmiast rozpoczniemy instancję obiektu usługi serwera za pomocą następującej komendy MQSC:

```
START SERVICE(S1)
```

3. Wyświetlany jest status procesu usługi serwera przy użyciu następującej komendy MQSC:

```
DISPLAY SVSTATUS(S1)
```

4. W tym przykładzie przedstawiono sposób zmiany obiektu usługi serwera i pobrania aktualizacji przez ręczne restartowanie procesu obsługi serwera. Obiekt usługi serwera jest zmieniany w taki sposób, że kolejka inicjujący jest określona jako JUPITER.INITIATION.QUEUE . Używana jest następująca komenda MQSC:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

Uwaga: Działająca usługa nie będzie odbierać żadnych aktualizacji definicji usługi, dopóki nie zostanie zrestartowana.

5. Proces usługi serwera zostanie zrestartowany, aby zmiana została pobrana z użyciem następujących komend MQSC:

```
STOP SERVICE(S1)
```

Następnie:

```
START SERVICE(S1)
```

Proces usługi serwera jest restartowany i pobiera zmiany wprowadzone w produkcie “4” na stronie 104.

Uwaga: Komenda MQSC STOP SERVICEmoże być używana tylko wtedy, gdy w definicji usługi jest określony argument STOPCMD.

Korzystanie z obiektu usługi komendy

W tym przykładzie przedstawiono sposób definiowania obiektu usługi komendy w celu uruchomienia programu, który zapisuje wpisy w dzienniku systemowym systemu operacyjnego, gdy menedżer kolejek jest uruchomiony lub zatrzymany.

1. Obiekt usługi komendy jest zdefiniowany przy użyciu następującej komendy MQSC:

```
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')
```

gdzie:

`logger` to komenda dostarczona przez system UNIX and Linux , która służy do zapisu w dzienniku systemowym.

+QMNAME+ jest znacznikiem reprezentującym nazwę menedżera kolejek.

Korzystanie z obiektu usługi komendy, gdy menedżer kolejek kończy się tylko

W tym przykładzie przedstawiono sposób definiowania obiektu usługi komendy w celu uruchomienia programu, który zapisuje wpisy w dzienniku systemowym systemu operacyjnego, gdy menedżer kolejek jest zatrzymany.

1. Obiekt usługi komendy jest zdefiniowany przy użyciu następującej komendy MQSC:

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

gdzie:

Program `logger` jest przykładowym programem dostarczonym z produktem WebSphere MQ , który może zapisywać wpisy w dzienniku systemowym systemu operacyjnego.

+QMNAME+ jest znacznikiem reprezentującym nazwę menedżera kolejek.

Więcej na temat przekazywania argumentów

W tym przykładzie przedstawiono sposób definiowania obiektu usługi serwera w celu uruchomienia programu o nazwie `runserv` , gdy uruchamiany jest menedżer kolejek.

Ten przykład jest zapisywany w postaci znaków separatora ścieżek w stylu systemu Windows .

Jednym z argumentów, które mają być przekazane do programu uruchamianego, jest łańcuch zawierający spację. Ten argument musi być przekazywany jako pojedynczy łańcuch. Aby to osiągnąć, należy użyć podwójnych cudzysłowów, jak pokazano w poniższej komendzie, w celu zdefiniowania obiektu usługi komendy:

1. Obiekt usługi serwera jest zdefiniowany za pomocą następującej komendy MQSC:

```
   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')
```

gdzie:

+QMNAME+ jest znacznikiem reprezentującym nazwę menedżera kolejek.

`"C:\Program Files\Tools\"` to łańcuch zawierający spację, która zostanie przekazana jako pojedynczy łańcuch.

Automatyczne uruchamianie usługi

W tym przykładzie przedstawiono sposób definiowania obiektu usługi serwera, który może być używany do automatycznego uruchamiania monitora wyzwalacza podczas uruchamiania menedżera kolejek.

1. Obiekt usługi serwera jest zdefiniowany za pomocą następującej komendy MQSC:

```
DEFINE SERVICE(TRIG_MON_START) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('runmqtrm') +
       STARTARG('-m +QMNAME+ -q +IQNAME+')
```

gdzie:

+QMNAME+ jest znacznikiem reprezentującym nazwę menedżera kolejek.

+IQNAME+ to zmienna środowiskowa zdefiniowana przez użytkownika w jednym z plików service.env , które reprezentują nazwę kolejki inicjuj.

Zarządzanie obiektami w celu wyzwolenia

Produkt WebSphere MQ umożliwia automatyczne uruchamianie aplikacji, gdy spełnione są określone warunki w kolejce. Użytkownik może na przykład chcieć uruchomić aplikację, gdy liczba komunikatów w kolejce osiągnie określoną liczbę. Ta funkcja jest nazywana *wyzwalaniem*. Konieczne jest zdefiniowanie obiektów, które obsługują wyzwalanie.

Wyzwalanie opisane szczegółowo w Uruchamianie aplikacji WebSphere MQ przy użyciu wyzwalaczy.

Definiowanie kolejki aplikacji na potrzeby wyzwalania

Kolejka aplikacji jest kolejką lokalną, która jest używana przez aplikacje do przesyłania komunikatów, przez interfejs MQI. Wyzwalanie wymaga zdefiniowania liczby atrybutów kolejki w kolejce aplikacji.

Samo wyzwolenie jest włączone przez atrybut *Trigger* (TRIGGER w komendach MQSC).
W tym przykładzie zdarzenie wyzwalające ma zostać wygenerowane, gdy w kolejce lokalnej MOTOR.INSURANCE.QUEUE, w następujący sposób:

```
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)
```

gdzie:

QLOCAL (MOTOR.INSURANCE.QUEUE)
Jest nazwą definiowanej kolejki aplikacji.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
Jest to nazwa definicji procesu definiująca aplikację, która ma zostać uruchomiona przez program monitora wyzwalacza.

MAXMSGL (2000)
Maksymalna długość komunikatów w kolejce.

DEFPSIST (YES)
Określa, że komunikaty w tej kolejce są domyślnie trwałe.

INITQ (MOTOR.INS.INIT.QUEUE)
Jest to nazwa kolejki inicjuj, w której menedżer kolejek ma umieścić komunikat wyzwalacza.

TRIGGER
Jest to wartość atrybutu wyzwalacza.

TRIGTYPE (DEPTH)
Określa, że zdarzenie wyzwalające jest generowane, gdy liczba komunikatów o wymaganym priorytecie (TRIGMPRI) osiągnie liczbę określoną w TRIGDPTH.

TRIGDPTH (100)
Liczba komunikatów wymaganych do wygenerowania zdarzenia wyzwalającego.

TRIGMPRI (5)
Jest to priorytet komunikatów, które mają być zliczane przez menedżera kolejek przy podejmowaniu decyzji o wygenerowaniu zdarzenia wyzwalającego. Zliczane są tylko komunikaty o priorytecie 5 lub wyższym.

Definiowanie kolejki inicjuj

Gdy wystąpi zdarzenie wyzwalające, menedżer kolejek umieszcza komunikat wyzwalacza w kolejce inicjuj. określonej w definicji kolejki aplikacji. Kolejki inicjujący nie mają żadnych ustawień specjalnych, ale można użyć następującej definicji kolejki lokalnej MOTOR.INS.INIT.QUEUE , aby uzyskać informacje na temat:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Definiowanie procesu

Aby utworzyć definicję procesu, należy użyć komendy DEFINE PROCESS. Definicja procesu definiuje aplikację, która ma być używana do przetwarzania komunikatów z kolejki aplikacji. Definicja kolejki aplikacji określa nazwę procesu, który ma zostać użyty, a tym samym wiąże kolejkę aplikacji z aplikacją, która ma być używana do przetwarzania jej komunikatów. Jest to realizowane za pomocą atrybutu PROCESS w kolejce aplikacji MOTOR.INSURANCE.QUEUE. Poniższa komenda MQSC definiuje wymagany proces, MOTOR.INSURANCE.QUOTE.PROCESS, zidentyfikowany w tym przykładzie:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

gdzie:

MOTOR.INSURANCE.QUOTE.PROCESS
Jest nazwą definicji procesu.

DESCR ('Insurance request message processing')
Opisuje program użytkowy, którego dotyczy ta definicja. Ten tekst jest wyświetlany, gdy używana jest komenda DISPLAY PROCESS. Może to pomóc w określeniu, co proces ma. Jeśli w łańcuchu używane są spacje, łańcuch należy ująć w pojedynczy cudzysłów.

APPLTYPE (UNIX)
Typ aplikacji, która ma zostać uruchomiona.

APPLICID ('/u/admin/test/IRMP01')
Jest nazwą pliku wykonywalnego aplikacji, który jest określony jako pełna nazwa pliku. W systemach Windows typowa wartość APPLICID będzie mieć wartość `c:\appl\test\irmp01.exe`.

USERDATA ('open, close, 235')
Jest to dane zdefiniowane przez użytkownika, które mogą być używane przez aplikację.

Wyświetlanie atrybutów definicji procesu

Użyj komendy DISPLAY PROCESS, aby sprawdzić wyniki definicji. Na przykład:

```
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)
```

Można również użyć komendy MQSC ALTER PROCESS w celu zmiany istniejącej definicji procesu oraz komendy DELETE PROCESS w celu usunięcia definicji procesu.

Administrowanie zdalnymi obiektami IBM WebSphere MQ

W tej sekcji opisano sposób administrowania obiektami produktu IBM WebSphere MQ w zdalnym menedżerze kolejek za pomocą komend MQSC oraz sposobu korzystania z obiektów kolejki zdalnej w celu sterowania miejscem docelowym komunikatów i komunikatów odpowiedzi.

W tej sekcji opisano:

- “Kanały, klastry i kolejkowanie zdalne” na stronie 108
- “Zdalne administrowanie z lokalnego menedżera kolejek” na stronie 110
- “Tworzenie lokalnej definicji kolejki zdalnej” na stronie 116
- “Używanie definicji kolejek zdalnych jako aliasów” na stronie 118
- “Konwersja danych między kodowanymi zestawami znaków” na stronie 119

Kanały, klastry i kolejkowanie zdalne

Menedżer kolejek komunikuje się z innym menedżerem kolejek, wysyłając komunikat i, jeśli jest to wymagane, odbierając odpowiedź. Odbiorcą menedżera kolejek może być:

- Na tym samym komputerze
- Na innym komputerze w tej samej lokalizacji (lub nawet po drugiej stronie świata)
- Uruchamianie na tej samej platformie co lokalny menedżer kolejek
- Uruchamianie na innej platformie obsługiwanej przez produkt WebSphere MQ

Te komunikaty mogą pochodzić z:

- Programy użytkowe napisane przez użytkownika, które przesyłają dane z jednego węzła do innego
- Aplikacje administracyjne napisane przez użytkownika, które korzystają z komend PCF lub MQAI
- Eksplorator IBM WebSphere MQ .
- Wysyłanie menedżerów kolejek:
  - Instrumentacja komunikatów zdarzeń do innego menedżera kolejek
  - Komendy MQSC wydane z komendy `runmqsc` w trybie pośrednim (w przypadku uruchamiania komend w innym menedżerze kolejek)

Zanim komunikat może zostać wysłany do zdalnego menedżera kolejek, lokalny menedżer kolejek wymaga mechanizmu wykrywania pojawiania się komunikatów i transportu ich składającego się z następujących elementów:

- Co najmniej jeden kanał
- Kolejka transmisji

- Inicjator kanału

W przypadku menedżera kolejek zdalnych do odebrania komunikatu wymagany jest obiekt nasłuchiwania.

Kanał jest jednokierunkową łączem komunikacyjnym między dwoma menedżerami kolejek i może przenosić komunikaty przeznaczone dla dowolnej liczby kolejek w zdalnym menedżerze kolejek.

Każdy koniec kanału ma osobną definicję. Na przykład, jeśli jeden koniec jest nadawcą lub serwerem, drugi koniec musi być odbiornikiem lub requesterem. Prosty kanał składa się z *definicji kanału nadawczego* na końcu lokalnego menedżera kolejek i *definicji kanału odbiorczego* w zdalnym menedżerze kolejek. Te dwie definicje muszą mieć taką samą nazwę i razem tworzą jeden kanał komunikatów.

Jeśli zdalny menedżer kolejek ma odpowiadać na komunikaty wysłane przez lokalny menedżer kolejek, należy skonfigurować drugi kanał w celu wysyłania odpowiedzi z powrotem do lokalnego menedżera kolejek.

Aby zdefiniować kanały, należy użyć komendy MQSC DEFINE CHANNEL. W tej sekcji przykłady odnoszące się do kanałów używają domyślnych atrybutów kanału, o ile nie określono inaczej.

Na każdym końcu kanału znajduje się agent kanału komunikatów (MCA), kontrolujący wysyłanie i odbieranie komunikatów. Agent MCA pobiera komunikaty z kolejki transmisji i umieszcza je na łączu komunikacyjnym między menedżerami kolejek.

Kolejka transmisji jest wyspecjalizowaną kolejką lokalną, która tymczasowo przechowuje komunikaty, zanim agent MCA pobiera je i wysyła do zdalnego menedżera kolejek. Nazwę kolejki transmisji należy określić w *definicji kolejki zdalnej*.

Można zezwolić agentowi MCA na przesyłanie komunikatów przy użyciu wielu wątków. Ten proces jest znany jako *pipelining*. Pipelining umożliwia agentowi MCA bardziej efektywne przesyłanie komunikatów, poprawiając wydajność kanałów. Sekcja Atrybuty kanałów zawiera szczegółowe informacje na temat sposobu konfigurowania kanału w celu korzystania z pipekresów.

W sekcji “Przygotowywanie kanałów i kolejek transmisji do zdalnego administrowania” na stronie 111 opisano sposób korzystania z tych definicji w celu skonfigurowania zdalnego administrowania.

Więcej informacji na temat konfigurowania rozproszonego kolejkowania w ogólności znajduje się w sekcji Rozproszone komponenty kolejkowania.

Zdalne administrowanie przy użyciu klastrów

W przypadku sieci WebSphere MQ korzystającej z rozproszonego kolejkowania każdy menedżer kolejek jest niezależny. Jeśli jeden menedżer kolejek musi wysyłać komunikaty do innego menedżera kolejek, musi zdefiniować kolejkę transmisji, kanał do zdalnego menedżera kolejek oraz definicję kolejki zdalnej dla każdej kolejki, do której ma być wysyłane komunikaty.

Klaster jest grupą menedżerów kolejek ustawionymi w taki sposób, że menedżery kolejek mogą komunikować się bezpośrednio z jedną siecią za pośrednictwem jednej sieci bez złożonej kolejki transmisji, kanału i definicji kolejek. Klastry mogą być łatwo skonfigurowane i zwykle zawierają menedżery kolejek, które są logicznie powiązane w pewien sposób i muszą współużytkować dane lub aplikacje. Nawet najmniejszy klaster redukuje koszty administrowania systemem.

Utworzenie sieci menedżerów kolejek w klastrze obejmuje mniejszą liczbę definicji niż utworzenie tradycyjnego rozproszonego środowiska kolejkowania. Dzięki mniejszej liczbie definicji można szybciej i łatwo skonfigurować lub zmienić sieć, a także zmniejszyć ryzyko popełnia błędu w definicjach.

Aby skonfigurować klaster, potrzebny jest jeden nadawca klastra (CLUSSDR) i jeden odbiornik klastra (CLUSRCVR) dla każdego menedżera kolejek. Nie są potrzebne żadne definicje kolejek transmisji ani definicje kolejek zdalnych. Zasady zdalnego administrowania są takie same, gdy używane są w klastrze, ale same definicje są znacznie uproszczone.

Więcej informacji na temat klastrów, ich atrybutów i sposobu ich konfigurowania zawiera sekcja Klastry menedżera kolejek.

Zdalne administrowanie z lokalnego menedżera kolejek

Ta sekcja zawiera informacje na temat administrowania zdalnym menedżerem kolejek z lokalnego menedżera kolejek za pomocą komend MQSC i PCF.

Przygotowanie kolejek i kanałów jest zasadniczo takie samo w przypadku komend MQSC i PCF. W tej sekcji przykłady ilustrują komendy MQSC, ponieważ są one łatwiejsze do zrozumienia. Aby uzyskać więcej informacji na temat pisania programów administracyjnych za pomocą komend PCF, patrz “Korzystanie z formatów komend programowalnych” na stronie 10.

Komendy MQSC wysyłane są do zdalnego menedżera kolejek interaktywnie lub z pliku tekstowego zawierającego komendy. Zdalny menedżer kolejek może znajdować się na tym samym komputerze lub na innym komputerze, zazwyczaj na innym komputerze. Menedżery kolejek można administrować zdalnie w innych środowiskach WebSphere MQ , w tym w systemach UNIX and Linux , systemach Windows , IBM ii z/OS.

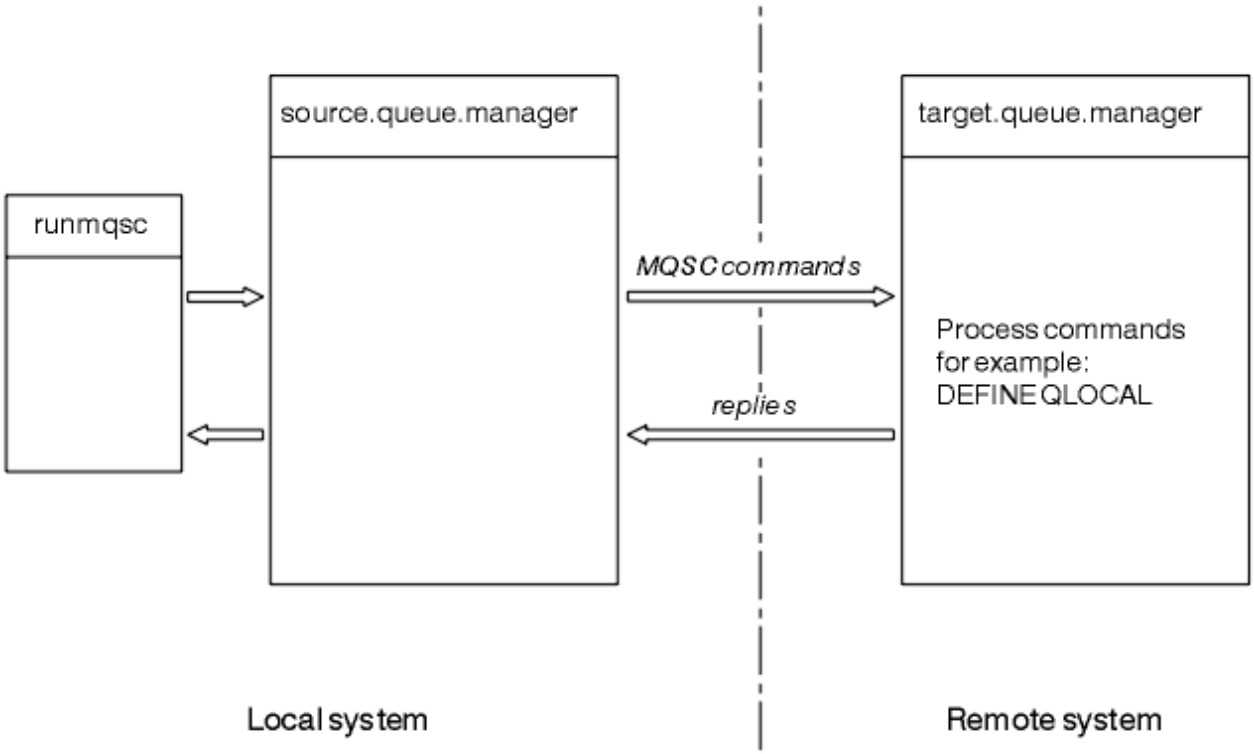
Aby zaimplementować zdalne administrowanie, należy utworzyć konkretne obiekty. Jeśli nie są wymagane specjalne wymagania, wartości domyślne (na przykład dla maksymalnej długości komunikatu) są wystarczające.

Przygotowywanie menedżerów kolejek do zdalnego administrowania

Informacje na temat używania komend MQSC do przygotowywania menedżerów kolejek do zdalnego administrowania.

W programie Rysunek 17 na stronie 111 jest wyświetlana konfiguracja menedżerów kolejek i kanałów, które są potrzebne do zdalnego administrowania za pomocą komendy **runmqsc** . Obiekt `source.queue.manager` jest źródłowym menedżerem kolejek, z którego można wydawać komendy MQSC i do których zwracane są wyniki tych komend (komunikaty operatora). Obiekt `target.queue.manager` to nazwa docelowego menedżera kolejek, który przetwarza komendy i generuje dowolne komunikaty operatora.

Uwaga: Jeśli używana jest opcja **runmqsc** z opcją -w, to `source.queue.manager` ***musi*** być domyślnym menedżerem kolejek. Więcej informacji na temat tworzenia menedżera kolejek można znaleźć w sekcji crtmqm.



Rysunek 17. Zdalne administrowanie za pomocą komend MQSC

W obu systemach, jeśli nie zostało to jeszcze wykonane:

- Utwórz menedżer kolejek i obiekty domyślne, używając komendy `crtmqm` .
- Uruchom menedżera kolejek za pomocą komendy `strmqm` .

W docelowym menedżerze kolejek:

- Kolejka komend SYSTEM.ADMIN.COMMAND.QUEUE, musi być obecna. Ta kolejka jest tworzona domyślnie, gdy tworzony jest menedżer kolejek.

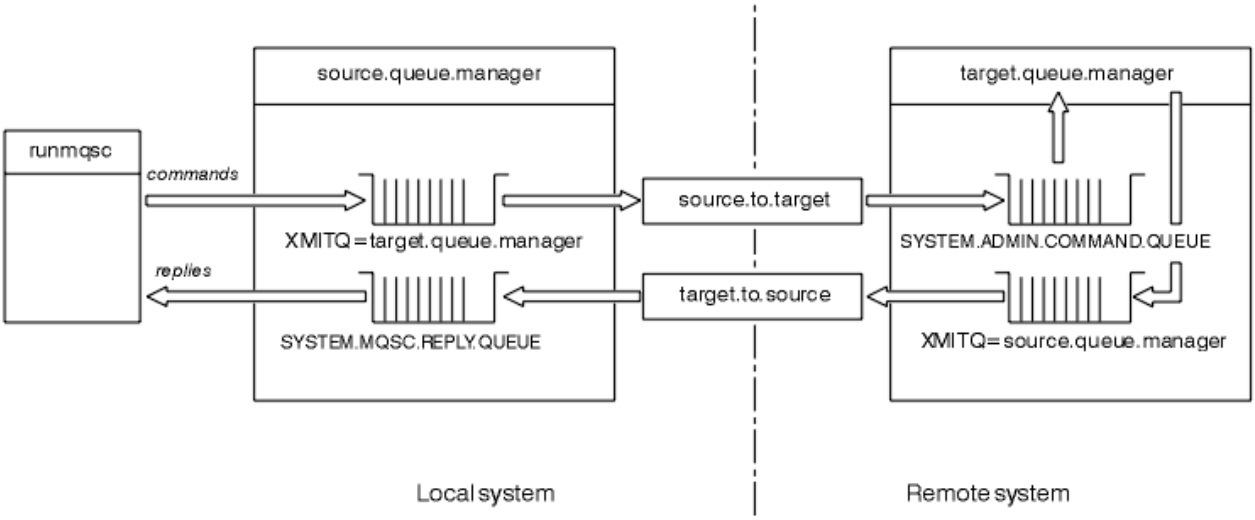
Komendy te należy uruchamiać lokalnie lub za pośrednictwem narzędzia sieciowego, takiego jak Telnet.

Przygotowywanie kanałów i kolejek transmisji do zdalnego administrowania

Jak używać komend MQSC do przygotowywania kanałów i kolejek transmisji dla zdalnego administrowania.

Aby zdalnie uruchamiać komendy MQSC, należy skonfigurować dwa kanały, po jednym dla każdego kierunku i powiązane z nimi kolejki transmisji. W tym przykładzie założono, że jako typ transportu używany jest protokół TCP/IP oraz że jest on używany jako adres TCP/IP.

Kanał `source.to.target` jest przeznaczony do wysyłania komend MQSC z źródłowego menedżera kolejek do docelowego menedżera kolejek. Jego nadawca znajduje się w `source.queue.manager` , a jego odbiornik znajduje się w `target.queue.manager`. Kanał `target.to.source` służy do zwracania danych wyjściowych komend i wszystkich komunikatów operatora, które są generowane dla źródłowego menedżera kolejek. Należy również zdefiniować kolejkę transmisji dla każdego kanału. Ta kolejka jest kolejką lokalną, której nadano nazwę odbierającego menedżera kolejek. Nazwa kolejki XMITQ musi być zgodna z nazwą zdalnego menedżera kolejek w celu zdalnego administrowania do pracy, chyba że używany jest alias menedżera kolejek. Rysunek 18 na stronie 112 podsumowuje tę konfigurację.



Rysunek 18. Konfigurowanie kanałów i kolejek do zdalnego administrowania

Więcej informacji na temat konfigurowania kanałów zawiera sekcja Łączenie aplikacji za pomocą rozproszonego kolejkowania .

Definiowanie kanałów, programów nasłuchujących i kolejek transmisji

W źródłowym menedżerze kolejek (`source.queue.manager`) wykonaj następujące komendy MQSC, aby zdefiniować kanały, nasłuchiwanie i kolejkę transmisji:

1. Zdefiniuj kanał nadawczy w źródłowym menedżerze kolejek:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. Zdefiniuj kanał odbiorczy w źródłowym menedżerze kolejek:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Zdefiniuj program nasłuchujący w źródłowym menedżerze kolejek:

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. Zdefiniuj kolejkę transmisji w źródłowym menedżerze kolejek:

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

Wykonaj następujące komendy w docelowym menedżerze kolejek (`target.queue.manager`), aby utworzyć kanały, nasłuchiwanie i kolejkę transmisji:

1. Zdefiniuj kanał nadawczy w docelowym menedżerze kolejek:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. Zdefiniuj kanał odbiorczy w docelowym menedżerze kolejek:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Zdefiniuj program nasłuchujący w docelowym menedżerze kolejek:

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. Zdefiniuj kolejkę transmisji w docelowym menedżerze kolejek:

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

Uwaga: Nazwy połączeń TCP/IP określone dla atrybutu CONNAME w definicjach kanału nadawczego służą tylko do ilustracji. Jest to nazwa sieciowa komputera na *innym* końcu połączenia. Należy użyć wartości odpowiednich dla danej sieci.

Uruchamianie programów nasłuchujących i kanałów

W jaki sposób można używać komend MQSC do uruchamiania programów nasłuchujących i kanałów.

Uruchom oba programy nasłuchujące, korzystając z następujących komend MQSC:

1. Uruchom program nasłuchujący w źródłowym menedżerze kolejek, `source.queue.manager`, wydając następującą komendę MQSC:

```
START LISTENER ('source.queue.manager')
```

2. Uruchom program nasłuchujący w docelowym menedżerze kolejek, `target.queue.manager`, wydając następującą komendę MQSC:

```
START LISTENER ('target.queue.manager')
```

Uruchom oba kanały wysyłające za pomocą następujących komend MQSC:

1. Uruchom kanał nadawczy w źródłowym menedżerze kolejek, `source.queue.manager`, wydając następującą komendę MQSC:

```
START CHANNEL ('source.to.target')
```

2. Uruchom kanał nadawczy w docelowym menedżerze kolejek, `target.queue.manager`, wydając następującą komendę MQSC:

```
START CHANNEL ('target.to.source')
```

Automatyczna definicja kanałów

Można włączyć automatyczną definicję definicji połączenia odbiorcy i serwera, aktualizując obiekt menedżera kolejek za pomocą komendy MQSC ALTER QMGR (lub komendy PCF Zmiana menedżera kolejek).

Jeśli program WebSphere MQ odbiera przychodzące żądanie przyłączenia i nie może znaleźć odpowiedniego kanału odbiorczego lub kanału połączenia z serwerem, tworzy kanał automatycznie. Definicje automatyczne są oparte na dwóch definicjach domyślnych dostarczanych razem z produktem WebSphere MQ: SYSTEM.AUTO.RECEIVER i SYSTEM.AUTO.SVRCONN.

Więcej informacji na temat automatycznego tworzenia definicji kanałów znajduje się w sekcji Przygotowywanie kanałów. Więcej informacji na temat automatycznego definiowania kanałów dla klastrów zawiera sekcja Auto-definition of cluster channels(Automatyczna definicja kanałów klastra).

Zarządzanie serwerem komend w celu zdalnego administrowania

Informacje na temat uruchamiania, zatrzymywania i wyświetlania statusu serwera komend. Serwer komend jest obowiązkowy dla wszystkich funkcji administracyjnych związanych z komendami PCF, MQAI, a także dla zdalnego administrowania.

Z każdym menedżerem kolejek może być powiązany serwer komend. Serwer komend przetwarza wszystkie komendy przychodzące ze zdalnych menedżerów kolejek lub komendy PCF z aplikacji. Przedstawia on komendy do menedżera kolejek w celu przetworzenia i zwraca kod zakończenia lub komunikat operatora w zależności od pochodzenia komendy.

Uwaga: W przypadku zdalnego administrowania upewnij się, że docelowy menedżer kolejek jest uruchomiony. W przeciwnym razie komunikaty zawierające komendy nie mogą opuścić menedżera kolejek, z którego są one wydawane. Zamiast tego komunikaty te są kolejkowane w lokalnej kolejce transmisji, która służy do obsługi menedżera kolejek zdalnych. Należy unikać tej sytuacji.

Istnieją oddzielne komendy sterujące służące do uruchamiania i zatrzymywania serwera komend. Jeśli serwer komend jest uruchomiony, użytkownicy produktu WebSphere MQ for Windows lub WebSphere MQ for Linux (platformyx86 i x86-64) mogą wykonywać operacje opisane w poniższych sekcjach za pomocą programu IBM WebSphere MQ Explorer. Więcej informacji na ten temat zawiera sekcja “Administrowanie przy użyciu IBM WebSphere MQ Explorer” na stronie 58.

Uruchamianie serwera komend

W zależności od wartości atrybutu menedżera kolejek (*SCMDSERV*) serwer komend jest uruchamiany automatycznie podczas uruchamiania menedżera kolejek lub musi być uruchamiany ręcznie. Wartość atrybutu menedżera kolejek można zmienić za pomocą komendy MQSC `ALTER QMGR` , określając parametr SCMDSERV. Domyślnie serwer komend jest uruchamiany automatycznie.

Jeśli parametr *SCMDSERV* jest ustawiony na wartość MANUAL, uruchom serwer komend przy użyciu komendy:

```
strmqcsv saturn.queue.manager
```

gdzie `saturn.queue.manager` jest menedżerem kolejek, dla którego uruchamiany jest serwer komend.

Wyświetlanie statusu serwera komend

W przypadku zdalnego administrowania należy upewnić się, że serwer komend w docelowym menedżerze kolejek jest uruchomiony. Jeśli nie jest uruchomiony, komendy zdalne nie mogą być przetwarzane. Wszystkie komunikaty zawierające komendy są kolejkowane w kolejce komend menedżera kolejek docelowych.

Aby wyświetlić status serwera komend dla menedżera kolejek, wprowadź następującą komendę MQSC:

```
DISPLAY QMSTATUS CMDSERV
```

Zatrzymywanie serwera komend

Aby zakończyć działanie serwera komend uruchomionego w poprzednim przykładzie, należy użyć następującej komendy:

```
endmqcsv saturn.queue.manager
```

Serwer komend można zatrzymać na dwa sposoby:

- W przypadku zatrzymania kontrolowanego należy użyć komendy endmqcsv z opcją -c, która jest wartością domyślną.
- Aby zatrzymać natychmiastowe zatrzymanie, użyj komendy endmqcsv z opcją -i.

Uwaga: Zatrzymanie menedżera kolejek powoduje również zakończenie powiązanego z nim serwera komend.

Wydawanie komend MQSC w zdalnym menedżerze kolejek

W celu uruchomienia komend MQSC w zdalnym menedżerze kolejek można użyć konkretnego formularza komendy **runmqsc** .

Serwer komend ***musi*** być uruchomiony w docelowym menedżerze kolejek, jeśli ma być zdalnie przetwarzający komendy MQSC. (Nie jest to konieczne w menedżerze kolejek źródłowych). Informacje na temat uruchamiania serwera komend w menedżerze kolejek zawiera sekcja “Zarządzanie serwerem komend w celu zdalnego administrowania” na stronie 114.

W źródłowym menedżerze kolejek można następnie interaktywnie uruchamiać komendy MQSC w trybie pośrednim, wpisując:

```
runmqsc -w 30 target.queue.manager
```

Ta postać komendy runmqsc z opcją -w powoduje uruchomienie komend MQSC w trybie pośrednim, w których komendy są umieszczane (w zmodyfikowanym formacie) w kolejce wejściowej serwera komend i wykonywane w kolejności.

W przypadku wpisania komendy MQSC jest ona przekierowana do zdalnego menedżera kolejek, w tym przypadku, target.queue.manager. Limit czasu jest ustawiony na 30 sekund. Jeśli odpowiedź nie zostanie odebrana w ciągu 30 sekund, na lokalnym (źródłowym) menedżerze kolejek zostanie wygenerowany następujący komunikat:

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Po zatrzymaniu wydawania komend MQSC lokalny menedżer kolejek wyświetla wszystkie odpowiedzi z przekroczonym limitem czasu, które przybyły i odrzuciły wszelkie dalsze odpowiedzi.

Źródłowy menedżer kolejek jest domyślnie ustawiony na domyślny menedżer kolejek lokalnych. Jeśli w komendzie **runmqsc** zostanie podana opcja -m *LocalQmgrName* , można kierować komendy, które mają zostać wydane za pomocą dowolnego lokalnego menedżera kolejek.

W trybie pośrednim można również uruchomić plik komend MQSC w zdalnym menedżerze kolejek. Na przykład:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

gdzie mycomds.in jest plikiem zawierającym komendy MQSC, a report.out jest plikiem raportu.

Sugerowana metoda zdalnego wydawania komend

W przypadku wydawania komend w zdalnym menedżerze kolejek należy rozważyć zastosowanie następującego podejścia:

1. Umieść komendy MQSC, które mają być uruchamiane w systemie zdalnym w pliku komend.
2. Sprawdź, czy komendy MQSC są lokalnie, poprzez określenie opcji -v w komendzie runmqsc .

   Programu runmqsc nie można używać do weryfikowania komend MQSC w innym menedżerze kolejek.
3. Sprawdź, czy plik komend działa lokalnie bez błędów.
4. Uruchom plik komend w systemie zdalnym.

W przypadku problemów z używaniem komend MQSC

Jeśli masz problemy z uruchamiam zdalnie komendy MQSC, upewnij się, że masz:

- Uruchomiono serwer komend w docelowym menedżerze kolejek.
- Zdefiniowano poprawną kolejkę transmisji.
- Zdefiniowano dwa końce kanałów komunikatów dla obu tych kanałów:
  - Kanał, wzdłuż którego wysyłane są komendy.
  - Kanał, w którym mają zostać zwrócone odpowiedzi.
- W definicji kanału podano poprawną nazwę połączenia (CONNAME).
- Uruchomiono programy nasłuchujące przed rozpoczęciem kanałów komunikatów.
- Sprawdź, czy interwał odłączania nie utracił ważności, na przykład, jeśli kanał został uruchomiony, ale po pewnym czasie został zamknięty. Jest to szczególnie ważne w przypadku ręcznego uruchomienia kanałów.
- Wysłane żądania z menedżera kolejek źródłowych, które nie mają sensu w docelowym menedżerze kolejek (na przykład żądania zawierające parametry, które nie są obsługiwane w zdalnym menedżerze kolejek).

Patrz także “Rozwiązywanie problemów za pomocą komend MQSC” na stronie 83.

Tworzenie lokalnej definicji kolejki zdalnej

Lokalna definicja kolejki zdalnej jest definicją w lokalnym menedżerze kolejek, która odwołuje się do kolejki w zdalnym menedżerze kolejek.

Nie trzeba definiować kolejki zdalnej z pozycji lokalnej, ale jej zaletą jest to, że aplikacje mogą odwoływać się do zdalnej kolejki za pomocą jej lokalnie zdefiniowanej nazwy, zamiast określać nazwę kwalifikowaną za pomocą identyfikatora menedżera kolejek, w którym znajduje się zdalna kolejka.

Informacje o tym, jak działają lokalne definicje kolejek zdalnych

Aplikacja łączy się z lokalnym menedżerem kolejek, a następnie wywołuje wywołanie MQOPEN . W wywołaniu otwartym podana nazwa kolejki jest nazwą zdalnej definicji kolejki w lokalnym menedżerze kolejek. Definicja kolejki zdalnej dostarcza nazwy kolejki docelowej, docelowego menedżera kolejek i opcjonalnie kolejki transmisji. Aby umieścić komunikat w kolejce zdalnej, aplikacja wysyła wywołanie MQPUT , określając uchwyt zwrócony z wywołania MQOPEN . Menedżer kolejek używa nazwy kolejki zdalnej i nazwy zdalnego menedżera kolejek w nagłówku transmisji na początku komunikatu. Informacje te są używane do kierowania komunikatu do jego poprawnego miejsca docelowego w sieci.

Administrator może sterować miejscem docelowym komunikatu, modyfikując definicję kolejki zdalnej.

W poniższym przykładzie pokazano, w jaki sposób aplikacja umieszcza komunikat w kolejce, której właścicielem jest zdalny menedżer kolejek. Aplikacja łączy się z menedżerem kolejek, na przykład `saturn.queue.manager`. Właścicielem kolejki docelowej jest inny menedżer kolejek.

W wywołaniu programu MQOPEN , aplikacja określa następujące pola:

Wartość pola	Opis
ObjectName CYAN.REMOTE.QUEUE	Określa lokalną nazwę obiektu kolejki zdalnej. Definiuje kolejkę docelową i docelowy menedżer kolejek.
ObjectType (Kolejka)	Identyfikuje ten obiekt jako kolejkę.

Wartość pola	Opis
ObjectQmgrName Puste lub `saturn.queue.manager`	To pole jest opcjonalne. Jeśli ta wartość jest pusta, przyjmowana jest nazwa lokalnego menedżera kolejek. (Jest to menedżer kolejek, w którym istnieje definicja kolejki zdalnej).

Następnie aplikacja wydaje wywołanie MQPUT w celu umieszczenia komunikatu w tej kolejce.

W lokalnym menedżerze kolejek można utworzyć lokalną definicję kolejki zdalnej przy użyciu następujących komend MQSC:

```
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)
```

gdzie:

QREMOTE (CYAN.REMOTE.QUEUE)
Określa lokalną nazwę obiektu kolejki zdalnej. Jest to nazwa, którą aplikacje połączone z tym menedżerem kolejek muszą określić w wywołaniu MQOPEN , aby otworzyć kolejkę AUTOMOBILE.INSURANCE.QUOTE.QUEUE w zdalnym menedżerze kolejek `jupiter.queue.manager`.

DESCR ('Queue for auto insurance requests from the branches')
Udostępnia dodatkowy tekst opisujący korzystanie z kolejki.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Określa nazwę kolejki docelowej w zdalnym menedżerze kolejek. Jest to rzeczywista kolejka docelowa dla komunikatów wysyłanych przez aplikacje, które określają nazwę kolejki CYAN.REMOTE.QUEUE. Kolejka AUTOMOBILE.INSURANCE.QUOTE.QUEUE musi być zdefiniowana jako kolejka lokalna w zdalnym menedżerze kolejek.

RQMNAME (jupiter.queue.manager)
Określa nazwę zdalnego menedżera kolejek, do którego należy kolejka docelowa AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

XMITQ (INQUOTE.XMIT.QUEUE)
Określa nazwę kolejki transmisji. Jest to opcjonalne. Jeśli nazwa kolejki transmisji nie jest określona, używana jest kolejka o takiej samej nazwie, jak nazwa zdalnego menedżera kolejek.

W obu przypadkach odpowiednia kolejka transmisji musi być zdefiniowana jako kolejka lokalna z atrybutem *Usage* , który określa, że jest to kolejka transmisji (USAGE (XMITQ) w komendach MQSC).

Alternatywny sposób umieszczania komunikatów w kolejce zdalnej

Użycie lokalnej definicji kolejki zdalnej nie jest jedynym sposobem umieszczania komunikatów w kolejce zdalnej. Aplikacje mogą określać pełną nazwę kolejki, w tym nazwę zdalnego menedżera kolejek, jako część wywołania MQOPEN . W takim przypadku lokalna definicja kolejki zdalnej nie jest wymagana. Oznacza to jednak, że aplikacje muszą znać nazwę zdalnego menedżera kolejek w czasie wykonywania albo muszą mieć dostęp do niej.

Korzystanie z innych komend z kolejkami zdalnymi

Za pomocą komend MQSC można wyświetlać lub zmieniać atrybuty obiektu kolejki zdalnej lub można usunąć obiekt kolejki zdalnej. Na przykład:

- Aby wyświetlić atrybuty kolejki zdalnej:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- Aby zmienić kolejkę zdalną w celu włączenia operacji put. Nie ma to wpływu na kolejkę docelową. Tylko aplikacje, które określają tę kolejkę zdalną:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- Aby usunąć tę kolejkę zdalną. Nie ma to wpływu na kolejkę docelową, a tylko jej definicję lokalną:

```
  DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Uwaga: Usunięcie kolejki zdalnej powoduje usunięcie tylko lokalnej reprezentacji kolejki zdalnej. Nie należy usuwać samej kolejki zdalnej ani żadnych komunikatów na jej temat.

Definiowanie kolejki transmisji

Kolejka transmisji jest kolejką lokalną, która jest używana, gdy menedżer kolejek przekazuje komunikaty do zdalnego menedżera kolejek za pośrednictwem kanału komunikatów.

Kanał udostępnia jednokierunkowy odsyłacz do zdalnego menedżera kolejek. Komunikaty są kolejkowane w kolejce transmisji do momentu, gdy kanał będzie mógł je zaakceptować. Podczas definiowania kanału należy określić nazwę kolejki transmisji na końcu wysyłającego kanału komunikatów.

Atrybut USAGE komendy MQSC definiuje, czy kolejka jest kolejką transmisji, czy normalną kolejką.

Domyślne kolejki transmisji

Gdy menedżer kolejek wysyła komunikaty do zdalnego menedżera kolejek, identyfikuje kolejkę transmisji przy użyciu następującej sekwencji:

1. Kolejka transmisji określona w atrybucie XMITQ lokalnej definicji kolejki zdalnej.
2. Kolejka transmisji o tej samej nazwie, co docelowy menedżer kolejek. (Ta wartość jest wartością domyślną dla kolejki XMITQ lokalnej definicji kolejki zdalnej).
3. Kolejka transmisji określona w atrybucie DEFXMITQ lokalnego menedżera kolejek.

Na przykład następująca komenda MQSC tworzy domyślną kolejkę transmisji w systemie `source.queue.manager` dla komunikatów wybieranych do produktu `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Aplikacje mogą umieszczać komunikaty bezpośrednio w kolejce transmisji lub pośrednio przez definicję kolejki zdalnej. Patrz także “Tworzenie lokalnej definicji kolejki zdalnej” na stronie 116.

Używanie definicji kolejek zdalnych jako aliasów

Oprócz lokalizacji kolejki w innym menedżerze kolejek można również użyć lokalnej definicji kolejki zdalnej dla aliasów menedżera kolejek i aliasów kolejek zwrotnych. Oba typy aliasów są tłumaczane za pomocą lokalnej definicji kolejki zdalnej. Aby komunikat dotarł do miejsca docelowego, należy skonfigurować odpowiednie kanały.

Aliasy menedżera kolejek

Alias jest procesem, za pomocą którego nazwa docelowego menedżera kolejek, określona w komunikacie, jest modyfikowana przez menedżer kolejek na trasie komunikatu.Aliasy menedżera kolejek są ważne, ponieważ można ich używać do sterowania miejscem docelowym komunikatów w sieci menedżerów kolejek.

W tym celu należy zmienić definicję kolejki zdalnej w menedżerze kolejek w punkcie kontroli. Aplikacja wysyłający nie ma informacji o tym, że określona nazwa menedżera kolejek jest aliasem.

Więcej informacji na temat aliasów menedżera kolejek zawiera sekcja Czym są aliasy?.

Aliasy kolejek zwrotnych

Opcjonalnie aplikacja może określić nazwę kolejki zwrotnej, gdy w kolejce umieszcza *komunikat żądania* .

Jeśli aplikacja, która przetwarza komunikat, wyodrębnia nazwę kolejki odpowiedzi, wie, gdzie wysłać *komunikat odpowiedzi*(jeśli jest to wymagane).

Alias kolejki odpowiedzi to proces, za pomocą którego kolejka odpowiedzi określona w komunikacie żądania jest zmieniana przez menedżer kolejek na trasie komunikatu. Aplikacja wysyłający nie jest świadoma, że podana nazwa kolejki odpowiedzi jest aliasem.

Alias kolejki odpowiedzi pozwala zmienić nazwę kolejki odpowiedzi i opcjonalnie jej menedżera kolejek. To z kolei umożliwia kontrolowanie, która trasa jest używana dla komunikatów odpowiedzi.

Więcej informacji na temat komunikatów żądań, komunikatów odpowiedzi i kolejek odpowiedzi można znaleźć w sekcji Typy komunikatów i Kolejka odpowiedzi i menedżer kolejek.

Więcej informacji na temat aliasów kolejek odpowiedzi można znaleźć w temacie Aliasy i aliasy kolejki odpowiedzi .

Konwersja danych między kodowanymi zestawami znaków

Dane komunikatów w definiowanych formatach WebSphere MQ (zwane również *formatami wbudowanymi*) mogą być przekształcane przez menedżera kolejek z jednego kodowanego zestawu znaków na inny, pod warunkiem że oba zestawy znaków odnoszą się do jednego języka lub grupy podobnych języków.

Na przykład, konwersja między kodowanymi zestawami znaków z identyfikatorami (CCSID) 850 i 500 jest obsługiwana, ponieważ oba mają zastosowanie do języków zachodnioeuropejskich.

W przypadku konwersji znaków w kodzie EBCDIC nowej linii (NL) na format ASCII, należy zapoznać się z wszystkimi menedżerami kolejek .

Obsługiwane konwersje są zdefiniowane w sekcji Konwersja danych.

Gdy menedżer kolejek nie może przekształcić komunikatów w wbudowanych formatach

Menedżer kolejek nie może automatycznie konwertować komunikatów we wbudowanych formatach, jeśli ich identyfikatory CCSID reprezentują różne grupy języków narodowych. Na przykład konwersja między identyfikatorem CCSID 850 i identyfikatorem CCSID 1025 (który jest zestawem znaków kodowanych EBCDIC dla języków przy użyciu skryptu cyrylicą) nie jest obsługiwana, ponieważ wiele znaków w jednym kodowanym zestawie znaków nie może być reprezentowane w drugim. Jeśli istnieje sieć menedżerów kolejek pracujących w różnych językach narodowych, a konwersja danych między niektórymi kodowanymi zestawami znaków nie jest obsługiwana, można włączyć konwersję domyślną. Domyślna konwersja danych jest opisana w sekcji “Domyślna konwersja danych” na stronie 120.

Plik ccsid.tbl

Plik ccsid.tbl jest używany do następujących celów:

- W produkcie WebSphere MQ for Windows rejestrowane są wszystkie obsługiwane zestawy kodowe.
- Na platformach AIX i HP-UX obsługiwane zestawy kodów są przechowywane wewnętrznie przez system operacyjny.
- W przypadku wszystkich innych platform UNIX and Linux obsługiwane zestawy kodów są przechowywane w tabelach konwersji udostępnianych przez produkt WebSphere MQ.

- Określa on wszystkie dodatkowe zestawy kodowe. Aby określić dodatkowe zestawy kodów, należy przeprowadzić edycję pliku ccsid.tbl (wskazówki dotyczące sposobu wykonania tej czynności znajdują się w pliku).
- Określa ona domyślną konwersję danych.

Informacje zapisane w pliku ccsid.tblmożna zaktualizować. Można to zrobić, jeśli na przykład w przyszłej wersji systemu operacyjnego obsługiwane są dodatkowe kodowane zestawy znaków.

In WebSphere MQ for Okna, ccsid.tbl is located in directory `C:\Program Files\IBM\WebSphere MQ\conv\table` by default.

W produkcie WebSphere MQ for UNIX and Linux system ccsid.tbl znajduje się w katalogu `/var/mqm/conv/table`.

Domyślna konwersja danych

Jeśli kanały są skonfigurowane między dwoma komputerami, w których konwersja danych nie jest normalnie obsługiwana, należy włączyć domyślną konwersję danych dla kanałów do pracy.

Aby włączyć domyślną konwersję danych, należy zmodyfikować plik ccsid.tbl w celu określenia domyślnego identyfikatora CCSID EBCDIC i domyślnego identyfikatora CCSID ASCII. Instrukcje dotyczące tego, jak to zrobić, są zawarte w pliku. Należy to zrobić na wszystkich komputerach, które będą połączone za pomocą kanałów. Zrestartuj menedżer kolejek, aby zmiany zostały uwzględnione.

Domyślny proces konwersji danych jest następujący:

- Jeśli konwersja między źródłowym i docelowym identyfikatorem CCSID nie jest obsługiwana, ale identyfikatory CCSID w środowisku źródłowym i docelowym są zarówno EBCDIC, jak i ASCII, dane znakowe są przekazywane do aplikacji docelowej bez konwersji.
- Jeśli jeden identyfikator CCSID reprezentuje zestaw znaków kodowanych w kodzie ASCII, a drugi reprezentuje zestaw znaków kodowanych EBCDIC, produkt WebSphere MQ przekształca dane przy użyciu domyślnych identyfikatorów CCSID konwersji danych zdefiniowanych w pliku ccsid.tbl.

Uwaga: Spróbuj ograniczyć przekształcanie znaków do tych, które mają te same wartości kodowe w kodowanym zestawie znaków określonym dla komunikatu i w domyślnym kodowanym zestawie znaków. Jeśli używany jest tylko zestaw znaków, który jest poprawny dla nazw obiektów produktu WebSphere MQ (zgodnie z definicją w sekcji Nazewnictwo obiektów IBM WebSphere MQ), to w ogólnym przypadku spełni to wymaganie. Wyjątki występują w przypadku identyfikatorów CCSID EBCDIC 290, 930, 1279 i 5026 używanych w Japonii, gdzie małe litery mają różne kody od tych, które są używane w innych identyfikatorach CCSID EBCDIC.

Przekształcanie komunikatów w formatach zdefiniowanych przez użytkownika

Menedżer kolejek nie może konwertować komunikatów w formatach zdefiniowanych przez użytkownika z jednego kodowanego zestawu znaków na inny. Jeśli zachodzi potrzeba konwersji danych w formacie zdefiniowanym przez użytkownika, należy podać wyjście konwersji danych dla każdego takiego formatu. Nie należy używać domyślnych identyfikatorów CCSID do przekształcania danych znakowych w formatach zdefiniowanych przez użytkownika. Więcej informacji na temat przekształcania danych w formatach zdefiniowanych przez użytkownika oraz zapisywania wyjść konwersji danych znajduje się w sekcji Zapisywanie wyjść konwersji danych.

Zmiana identyfikatora CCSID menedżera kolejek

Po użyciu atrybutu CCSID komendy ALTER QMGR w celu zmiany identyfikatora CCSID menedżera kolejek należy zatrzymać i zrestartować menedżer kolejek, aby upewnić się, że wszystkie uruchomione aplikacje, w tym serwer komend i programy kanałów, zostaną zatrzymane i zrestartowane.

Jest to konieczne, ponieważ wszystkie aplikacje działające w momencie zmiany identyfikatora CCSID menedżera kolejek nadal korzystają z istniejącego identyfikatora CCSID.

administrowanieIBM WebSphere MQ Telemetry

Produkt IBM WebSphere MQ Telemetry jest administrowany przy użyciu produktu IBM WebSphere MQ Explorer lub w wierszu komend. Za pomocą eksploratora można skonfigurować kanały pomiarowe, sterować usługą telemetryczną i monitorować klienty MQTT, które są połączone z produktem IBM WebSphere MQ. Skonfiguruj zabezpieczenia produktu IBM WebSphere MQ Telemetry przy użyciu usług JAAS, protokołu SSL i menedżera uprawnień do obiektów produktu IBM WebSphere MQ .

Administrowanie przy użyciu produktu IBM WebSphere MQ Explorer

Za pomocą eksploratora można skonfigurować kanały pomiarowe, sterować usługą telemetryczną i monitorować klienty MQTT, które są połączone z produktem IBM WebSphere MQ. Skonfiguruj zabezpieczenia produktu IBM WebSphere MQ Telemetry przy użyciu usług JAAS, protokołu SSL i menedżera uprawnień do obiektów produktu IBM WebSphere MQ .

Administrowanie za pomocą wiersza komend

Produkt IBM WebSphere MQ Telemetry można w pełni administrować w wierszu komend za pomocą komend IBM WebSphere MQ MQSC .

Dokumentacja produktu IBM WebSphere MQ Telemetry zawiera również przykładowe skrypty, które demonstrują podstawowe użycie aplikacji klienckiej MQ Telemetry Transport v3 Client.

Przed użyciem tych przykładów należy zapoznać się z przykładami w przykładowych programach IBM WebSphere MQ Telemetry w sekcji Tworzenie aplikacji dla produktu IBM WebSphere MQ Telemetry .

Pojęcia pokrewne

WebSphere MQ Telemetry

“Konfigurowanie rozproszonego kolejkowania w celu wysyłania komunikatów do klientów MQTT” na stronie 125
Aplikacje produktu IBM WebSphere MQ mogą wysyłać komunikaty klientów MQTT v3 przez publikowanie do subskrypcji utworzonej przez klienta lub bezpośrednio przez wysłanie komunikatu. Niezależnie od tego, która metoda jest używana, komunikat jest umieszczany na serwerze `SYSTEM.MQTT.TRANSMIT.QUEUE`i wysyłany do klienta przez usługę telemetryczną (MQXR). Istnieje wiele sposobów umieszczania komunikatu w produkcie `SYSTEM.MQTT.TRANSMIT.QUEUE`.

“Identyfikacja klienta MQTT, autoryzacja i uwierzytelnianie” na stronie 128

“Uwierzytelnianie kanału telemetrycznego przy użyciu protokołu SSL” na stronie 135

“Prywatność publikacji na kanałach telemetrycznych” na stronie 137

“Konfiguracja SSL klientów MQTT i kanałów telemetrycznych” na stronie 138

“Konfiguracja kanału telemetrycznego JAAS” na stronie 143
Skonfiguruj usługę JAAS w taki sposób, aby uwierzytelniała `Nazwę użytkownika` wysłanego przez klienta.

“IBM WebSphere MQ Demon telemetryczny dla pojęć związanych z urządzeniami” na stronie 145
Demon IBM WebSphere MQ Telemetry dla urządzeń jest zaawansowaną aplikacją kliencką MQTT V3 . Służy do przechowywania i przekazywania komunikatów z innych klientów MQTT. Łączy się on z produktem IBM WebSphere MQ , takim jak klient MQTT, ale można również połączyć z nim inne klienty MQTT.

Zadania pokrewne

“Konfigurowanie menedżera kolejek na potrzeby telemetrii w systemach Linux i AIX” na stronie 122
Wykonaj poniższe czynności, aby skonfigurować menedżer kolejek do uruchamiania produktu IBM WebSphere MQ Telemetry. Można uruchomić zautomatyzowaną procedurę, aby skonfigurować prostszą konfigurację, korzystając z obsługi IBM WebSphere MQ Telemetry dla produktu IBM WebSphere MQ Explorer.

“Konfigurowanie menedżera kolejek na potrzeby telemetrii w produkcie Windows” na stronie 123
Wykonaj poniższe czynności, aby skonfigurować menedżer kolejek do uruchamiania produktu IBM WebSphere MQ Telemetry. Można uruchomić zautomatyzowaną procedurę, aby skonfigurować prostszą konfigurację, korzystając z obsługi IBM WebSphere MQ Telemetry dla produktu IBM WebSphere MQ Explorer.

Odsyłacze pokrewne
Właściwości MQXR

Konfigurowanie menedżera kolejek na potrzeby telemetrii w systemach Linux i AIX

Wykonaj poniższe czynności, aby skonfigurować menedżer kolejek do uruchamiania produktu IBM WebSphere MQ Telemetry. Można uruchomić zautomatyzowaną procedurę, aby skonfigurować prostszą konfigurację, korzystając z obsługi IBM WebSphere MQ Telemetry dla produktu IBM WebSphere MQ Explorer.

Zanim rozpoczniesz

1. Informacje na temat instalowania produktu IBM WebSphere MQoraz opcji IBM WebSphere MQ Telemetry zawiera sekcja Instalowanie produktu IBM WebSphere MQ Telemetry .
2. Utwórz i uruchom menedżer kolejek. W tym zadaniu menedżer kolejek jest określany jako *qMgr* .
3. W ramach tego zadania można skonfigurować usługę telemetryczną (MQXR). Ustawienia właściwości MQXR są przechowywane w pliku właściwości specyficznym dla platformy: `mqxr_unix.properties`. Zwykle nie ma potrzeby bezpośredniego edytowania pliku właściwości MQXR, ponieważ prawie wszystkie ustawienia można konfigurować za pomocą komend administracyjnych MQSC lub MQ Explorer. Jeśli zdecydujesz się bezpośrednio na edycję pliku, zatrzymaj menedżer kolejek przed dokonaniem zmian. Patrz: właściwości MQXR.

O tym zadaniu

Obsługa IBM WebSphere MQ Telemetry dla produktu IBM WebSphere MQ Explorer zawiera kreatora i przykładową procedurę komendy `sampleMQM`. Konfigurują one początkową konfigurację przy użyciu identyfikatora użytkownika gościa. Patrz sekcja Sprawdzanie poprawności instalacji produktu IBM WebSphere MQ Telemetry przy użyciu programu IBM WebSphere MQ Explorer i programów przykładowychIBM WebSphere MQ Telemetry.

Aby ręcznie skonfigurować produkt IBM WebSphere MQ Telemetry przy użyciu różnych schematów autoryzacji, należy wykonać kroki opisane w tej czynności.

Procedura

1. Otwórz okno komend w katalogu przykładów telemetrycznych.

   Katalog przykładów telemetrycznych to `/opt/mqm/mqxr/samples`.
2. Utwórz kolejkę transmisji telemetrycznej.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc
   qMgr
   ```

   Gdy usługa telemetryczna (MQXR) jest uruchamiana po raz pierwszy, tworzy ona `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Jest on tworzony ręcznie w tym zadaniu, ponieważ produkt `SYSTEM.MQTT.TRANSMIT.QUEUE` musi istnieć przed uruchomieniem usługi telemetrycznej (MQXR), aby autoryzować dostęp do niej.
3. Ustaw domyślną kolejkę transmisji

   Gdy usługa telemetryczna (MQXR) jest uruchamiana po raz pierwszy, nie zmienia menedżera kolejek, aby `SYSTEM.MQTT.TRANSMIT.QUEUE` była domyślną kolejką transmisji.

   Aby uczynić `SYSTEM.MQTT.TRANSMIT.QUEUE` domyślną kolejką transmisji, zmień domyślną właściwość kolejki transmisji. Zmień właściwość za pomocą konsoli IBM WebSphere MQ Explorer lub za pomocą komendy w następującym przykładzie:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

Zmiana domyślnej kolejki transmisji może kolidować z istniejącą konfiguracją. Przyczyną zmiany domyślnej kolejki transmisji na SYSTEM.MQTT.TRANSMIT.QUEUE jest ułatwienie wysyłania komunikatów bezpośrednio do klientów MQTT. Bez zmiany domyślnej kolejki transmisji należy dodać definicję kolejki zdalnej dla każdego klienta, który odbiera komunikaty produktu IBM WebSphere MQ ; patrz “Bezpośrednie wysyłanie wiadomości do klienta” na stronie 127.

4. Aby utworzyć jeden lub więcej identyfikatorów użytkowników, należy wykonać procedurę w produkcie “Autoryzowanie klientów MQTT w celu uzyskania dostępu do obiektów produktu WebSphere MQ” na stronie 129 . Identyfikatory użytkowników mają uprawnienia do publikowania, subskrybowania i wysyłania publikacji do klientów MQTT.
5. Zainstaluj usługę telemetryczną (MQXR)

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   Patrz także przykładowy kod w produkcie Rysunek 19 na stronie 123.
6. Uruchom usługę

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   Usługa telemetryczna (MQXR) jest uruchamiana automatycznie po uruchomieniu menedżera kolejek.

   Jest on uruchamiany ręcznie w tym zadaniu, ponieważ menedżer kolejek jest już uruchomiony.
7. Korzystając z produktu IBM WebSphere MQ Explorer, skonfiguruj kanały telemetryczne, aby akceptować połączenia z klientami MQTT.

   Kanały telemetryczne muszą być skonfigurowane w taki sposób, aby ich tożsamości były jednym z identyfikatorów zdefiniowanych w kroku 4.

   Patrz także: DEFINE CHANNEL (MQTT)(DEFINIOWANIE KANAŁU).
8. Sprawdź konfigurację, uruchamiając przykładową klienta.

   Aby przykładowy klient współpracował z kanałem telemetryczny, kanał musi autoryzować klienta do publikowania, subskrybowania i odbierania publikacji. Przykładowy klient łączy się domyślnie z kanałem telemetrycznej na porcie 1883. Patrz także IBM WebSphere MQ Telemetry przykładowe programy.

Przykład

Rysunek 19 na stronie 123 wyświetla komendę **runmqsc** , aby ręcznie utworzyć SYSTEM.MQXR.SERVICE na serwerze 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')
```

Rysunek 19. installMQXRService_unix.mqsc

Konfigurowanie menedżera kolejek na potrzeby telemetrii w produkcie Windows

Wykonaj poniższe czynności, aby skonfigurować menedżer kolejek do uruchamiania produktu IBM WebSphere MQ Telemetry. Można uruchomić zautomatyzowaną procedurę, aby skonfigurować prostszą konfigurację, korzystając z obsługi IBM WebSphere MQ Telemetry dla produktu IBM WebSphere MQ Explorer.

Zanim rozpoczniesz

1. Informacje na temat instalowania produktu IBM WebSphere MQoraz opcji IBM WebSphere MQ Telemetry zawiera sekcja Instalowanie produktu IBM WebSphere MQ Telemetry .
2. Utwórz i uruchom menedżer kolejek. W tym zadaniu menedżer kolejek jest określany jako *qMgr* .
3. W ramach tego zadania można skonfigurować usługę telemetryczną (MQXR). Ustawienia właściwości MQXR są przechowywane w pliku właściwości specyficznym dla platformy: `mqxr_win.properties`. Zwykle nie ma potrzeby bezpośredniego edytowania pliku właściwości MQXR, ponieważ prawie wszystkie ustawienia można konfigurować za pomocą komend administracyjnych MQSC lub MQ Explorer. Jeśli zdecydujesz się bezpośrednio na edycję pliku, zatrzymaj menedżer kolejek przed dokonaniem zmian. Patrz: właściwości MQXR.

O tym zadaniu

Obsługa IBM WebSphere MQ Telemetry dla produktu IBM WebSphere MQ Explorer zawiera kreatora i przykładową procedurę komendy `sampleMQM`. Konfigurują one początkową konfigurację przy użyciu identyfikatora użytkownika gościa. Patrz sekcja Sprawdzanie poprawności instalacji produktu IBM WebSphere MQ Telemetry przy użyciu programu IBM WebSphere MQ Explorer i programów przykładowychIBM WebSphere MQ Telemetry.

Aby ręcznie skonfigurować produkt IBM WebSphere MQ Telemetry przy użyciu różnych schematów autoryzacji, należy wykonać kroki opisane w tej czynności.

Procedura

1. Otwórz okno komend w katalogu przykładów telemetrycznych.

   Katalog przykładów telemetrycznych to *WMQ program installation directory*`\mqxr\samples`.
2. Utwórz kolejkę transmisji telemetrycznej.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   Gdy usługa telemetryczna (MQXR) jest uruchamiana po raz pierwszy, tworzy ona `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Jest on tworzony ręcznie w tym zadaniu, ponieważ produkt `SYSTEM.MQTT.TRANSMIT.QUEUE` musi istnieć przed uruchomieniem usługi telemetrycznej (MQXR), aby autoryzować dostęp do niej.
3. Ustaw domyślną kolejkę transmisji

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   Rysunek 20. Ustaw domyślną kolejkę transmisji

   Gdy usługa telemetryczna (MQXR) jest uruchamiana po raz pierwszy, nie zmienia menedżera kolejek, aby `SYSTEM.MQTT.TRANSMIT.QUEUE` była domyślną kolejką transmisji.

   Aby uczynić `SYSTEM.MQTT.TRANSMIT.QUEUE` domyślną kolejką transmisji, zmień domyślną właściwość kolejki transmisji. Zmień właściwość za pomocą IBM WebSphere MQ Explorer lub za pomocą komendy w programie Rysunek 20 na stronie 124.

   Zmiana domyślnej kolejki transmisji może kolidować z istniejącą konfiguracją. Przyczyną zmiany domyślnej kolejki transmisji na `SYSTEM.MQTT.TRANSMIT.QUEUE` jest ułatwienie wysyłania komunikatów bezpośrednio do klientów MQTT. Bez zmiany domyślnej kolejki transmisji należy dodać definicję kolejki zdalnej dla każdego klienta, który odbiera komunikaty produktu IBM WebSphere MQ ; patrz “Bezpośrednie wysyłanie wiadomości do klienta” na stronie 127.
4. Aby utworzyć jeden lub więcej identyfikatorów użytkowników, należy wykonać procedurę w produkcie “Autoryzowanie klientów MQTT w celu uzyskania dostępu do obiektów produktu WebSphere MQ” na stronie 129 . Identyfikatory użytkowników mają uprawnienia do publikowania, subskrybowania i wysyłania publikacji do klientów MQTT.

5. Zainstaluj usługę telemetryczną (MQXR)

   ```
   type
   installMQXRService_win.mqsc | runmqsc qMgr
   ```

6. Uruchom usługę

   ```
   echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
   ```

   Usługa telemetryczna (MQXR) jest uruchamiana automatycznie po uruchomieniu menedżera kolejek.

   Jest on uruchamiany ręcznie w tym zadaniu, ponieważ menedżer kolejek jest już uruchomiony.

7. Korzystając z produktu IBM WebSphere MQ Explorer, skonfiguruj kanały telemetryczne, aby akceptować połączenia z klientami MQTT.

   Kanały telemetryczne muszą być skonfigurowane w taki sposób, aby ich tożsamości były jednym z identyfikatorów zdefiniowanych w kroku 4.

   Patrz także: DEFINE CHANNEL (MQTT)(DEFINIOWANIE KANAŁU).

8. Sprawdź konfigurację, uruchamiając przykładową klienta.

   Aby przykładowy klient współpracował z kanałem telemetryczny, kanał musi autoryzować klienta do publikowania, subskrybowania i odbierania publikacji. Przykładowy klient łączy się domyślnie z kanałem telemetrycznej na porcie 1883. Patrz także IBM WebSphere MQ Telemetry przykładowe programy.

Ręczne tworzenie SYSTEM.MQXR.SERVICE

Rysunek 21 na stronie 125 przedstawia komendę **runmqsc** , aby ręcznie utworzyć SYSTEM.MQXR.SERVICE na serwerze 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')
```

Rysunek 21. installMQXRService_win.mqsc

Konfigurowanie rozproszonego kolejkowania w celu wysyłania komunikatów do klientów MQTT

Aplikacje produktu IBM WebSphere MQ mogą wysyłać komunikaty klientów MQTT v3 przez publikowanie do subskrypcji utworzonej przez klienta lub bezpośrednio przez wysłanie komunikatu. Niezależnie od tego, która metoda jest używana, komunikat jest umieszczany na serwerze SYSTEM.MQTT.TRANSMIT.QUEUEi wysyłany do klienta przez usługę telemetryczną (MQXR). Istnieje wiele sposobów umieszczania komunikatu w produkcie SYSTEM.MQTT.TRANSMIT.QUEUE.

Publikowanie komunikatu w odpowiedzi na subskrypcję klienta MQTT

Usługa telemetryczna (MQXR) tworzy subskrypcję w imieniu klienta MQTT. Klient jest miejscem docelowym dla wszystkich publikacji, które są zgodne z subskrypcją wysłanej przez klienta. Usługi telemetryczne przekazuje zgodne publikacje z powrotem do klienta.

Klient MQTT jest połączony z produktem WebSphere MQ jako menedżer kolejek, a jego nazwa menedżera kolejek jest ustawiona na wartość `ClientIdentifier`. Miejscem docelowym publikacji, które mają zostać wysłane do klienta, jest kolejka transmisji SYSTEM.MQTT.TRANSMIT.QUEUE. Usługa telemetryczna przekazuje komunikaty w produkcie SYSTEM.MQTT.TRANSMIT.QUEUE do klientów MQTT, korzystając z nazwy docelowego menedżera kolejek jako klucza do konkretnego klienta.

Usługa telemetryczna (MQXR) otwiera kolejkę transmisji przy użyciu wartości `ClientIdentifier` jako nazwy menedżera kolejek. Usługa telemetryczna (MQXR) przekazuje uchwyt obiektu kolejki do wywołania `MQSUB` , aby przekazać publikacje, które są zgodne z subskrypcją klienta. W rozwiązaniu nazwy obiektu `ClientIdentifier` jest tworzona jako nazwa zdalnego menedżera kolejek, a kolejka transmisji musi być przetłumaczana na `SYSTEM.MQTT.TRANSMIT.QUEUE`. Przy użyciu standardowej rozdzielczości nazwy obiektu produktu WebSphere MQ produkt *`ClientIdentifier`* jest rozstrzygany w sposób opisany poniżej. Patrz sekcja Tabela 6 na stronie 126.

1. *`ClientIdentifier`* nie pasuje do niczego.

   `ClientIdentifier` jest nazwą zdalnego menedżera kolejek. Nazwa ta nie jest zgodna z nazwą menedżera kolejek lokalnych, aliasem menedżera kolejek lub nazwą kolejki transmisji.

   Nazwa kolejki nie jest zdefiniowana. Obecnie usługa telemetryczna (MQXR) ustawia `SYSTEM.MQTT.PUBLICATION.QUEUE` jako nazwę kolejki. Klient MQTT v3 nie obsługuje kolejek, więc przetłumaczana nazwa kolejki jest ignorowana przez klienta.

   Właściwość lokalnego menedżera kolejek, `Domyślna kolejka transmisji`, nazwa musi być ustawiona na `SYSTEM.MQTT.TRANSMIT.QUEUE`, co powoduje, że publikacja jest umieszczana na serwerze `SYSTEM.MQTT.TRANSMIT.QUEUE` , który ma być wysyłany do klienta.

2. *`ClientIdentifier`* jest zgodny z aliasem menedżera kolejek o nazwie *`ClientIdentifier`*.

   `ClientIdentifier` jest nazwą zdalnego menedżera kolejek. Jest ona zgodna z nazwą aliasu menedżera kolejek.

   Alias menedżera kolejek musi być zdefiniowany z *`ClientIdentifier`* jako nazwa zdalnego menedżera kolejek.

   Ustawienie nazwy kolejki transmisji w definicji aliasu menedżera kolejek nie jest konieczne, aby domyślna transmisja została ustawiona na wartość `SYSTEM.MQTT.TRANSMIT.QUEUE`.

Tabela 6. Tłumaczenie nazw aliasu menedżera kolejek MQTT

	Wejście		**Wyjście**		
`ClientIdentifier`	**Nazwa menedżera kolejek**	**Nazwa kolejki**	**Nazwa menedżera kolejek**	**Nazwa kolejki**	**Kolejka transmisji**
Nic nie odpowiada	*`ClientIdentifier`*	*niezdefiniowane*	*`ClientIdentifier`*	*niezdefiniowane*	Domyślna kolejka transmisji. `SYSTEM.MQTT.TRANSMIT.QUEUE`
Jest zgodny z aliasem menedżera kolejek o nazwie *`ClientIdentifier`*	*`ClientIdentifier`*	*niezdefiniowane*	*`ClientIdentifier`*	*niezdefiniowane*	`SYSTEM.MQTT.TRANSMIT.QUEUE`

Więcej informacji na temat tłumaczenia nazw zawiera sekcja Rozdzielczość nazw.

Każdy program WebSphere MQ może publikować w tym samym temacie. Publikacja jest wysyłana do subskrybentów, w tym do klientów MQTT v3 , którzy mają subskrypcję tematu.

Jeśli temat administracyjny został utworzony w klastrze, z atrybutem `CLUSTER(`*`clusterName`*`)`, dowolna aplikacja w klastrze może publikować do klienta, na przykład:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

Rysunek 22. Definiowanie tematu klastra w systemie Windows

Uwaga: Nie należy nadawać SYSTEM.MQTT.TRANSMIT.QUEUE atrybutu klastra.

Subskrybenci klienta MQTT i publikatorzy mogą łączyć się z różnymi menedżerami kolejek. Subskrybenci i publikatorzy mogą być częścią tego samego klastra lub mogą być połączone w hierarchii publikowania/subskrypcji. Publikacja jest dostarczana z publikatora do subskrybenta przy użyciu produktu WebSphere MQ.

Bezpośrednie wysyłanie wiadomości do klienta

Alternatywa dla klienta tworującego subskrypcję i otrzymującego publikację zgodną z tematem subskrypcji, bezpośrednio wysyłają komunikat do klienta MQTT v3 . Aplikacje klienckie MQTT V3 nie mogą wysyłać komunikatów bezpośrednio, ale inne aplikacje, takie jak aplikacje produktu WebSphere MQ , mogą być wysyłane.

Aplikacja WebSphere MQ musi znać identyfikator ClientIdentifier klienta MQTT v3 . Ponieważ klienty MQTT v3 nie mają kolejek, nazwa kolejki docelowej jest przekazywana do metody messageArrived klienta aplikacji MQTT v3 jako nazwa tematu. Na przykład w programie MQI utwórz deskryptor obiektu z klientem jako ObjectQmgrName(Nazwa menedżera obiektów):

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

Rysunek 23. Deskryptor obiektu MQI w celu wysłania komunikatu do miejsca docelowego klienta MQTT v3

Jeśli aplikacja jest napisana przy użyciu usługi JMS, należy utworzyć miejsce docelowe typu punkt z punktem, na przykład:

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

Rysunek 24. Miejsce docelowe JMS w celu wysłania komunikatu do klienta MQTT v3

Aby wysłać niezamówiony komunikat do klienta MQTT, należy użyć definicji kolejki zdalnej. Nazwa zdalnego menedżera kolejek musi zostać przetłumaczona na wartość ClientIdentifier klienta. Kolejka transmisji musi być przetłumaczana na SYSTEM.MQTT.TRANSMIT.QUEUE; patrz Tabela 7 na stronie 127. Nazwa kolejki zdalnej może być dowolna. Klient otrzymuje go jako łańcuch tematu.

Tabela 7. Rozstrzyganie nazwy zdalnej definicji kolejki klienta MQTT

Wejście		Wyjście		
Nazwa kolejki	**Nazwa menedżera kolejek**	**Nazwa kolejki**	**Nazwa menedżera kolejek**	**Kolejka transmisji**
Nazwa definicji kolejki zdalnej	Pusta lub lokalna nazwa menedżera kolejek	Nazwa kolejki zdalnej używana jako łańcuch tematu	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

Jeśli klient jest połączony, komunikat jest wysyłany bezpośrednio do klienta MQTT, który wywołuje metodę messageArrived . Patrz sekcja MetodamessageArrived.

Jeśli klient rozłączył się z trwałą sesją, komunikat jest zapisywany w produkcie SYSTEM.MQTT.TRANSMIT.QUEUE. Patrz sekcja Stateless i sesje stanowe MQTT . Jest on przekazywany do klienta po ponownym nawiązaniu połączenia przez klienta z sesją.

Jeśli wysyłany jest komunikat nietrwały, jest on wysyłany do klienta z "co najwyżej raz" jakością usługi, QoS=0. Jeśli komunikat trwały jest wysyłany bezpośrednio do klienta, domyślnie jest on wysyłany z jakością usługi "dokładnie raz", QoS=2. Ponieważ klient może nie mieć mechanizmu trwałości, klient może obniżyć jakość usługi, którą akceptuje w przypadku komunikatów wysyłanych bezpośrednio. Aby obniżyć jakość usługi w przypadku komunikatów wysyłanych bezpośrednio do klienta, należy wykonać subskrypcję tematu DEFAULT.QoS. Określ maksymalną jakość usługi, którą może obsługiwać klient.

Identyfikacja klienta MQTT, autoryzacja i uwierzytelnianie

Usługa telemetryczna (MQXR) publikuje lub subskrybuje tematy produktu WebSphere MQ w imieniu klientów MQTT za pomocą kanałów MQTT. Administrator produktu WebSphere MQ konfiguruje tożsamość kanału MQTT, która jest używana dla autoryzacji produktu WebSphere MQ . Administrator może zdefiniować wspólną tożsamość dla kanału lub użyć wartości Username lub ClientIdentifier klienta połączonego z kanałem.

Usługa telemetryczna (MQXR) może uwierzytelnić klienta przy użyciu nazwy użytkownika dostarczonej przez klienta lub za pomocą certyfikatu klienta. Nazwa użytkownika jest uwierzytelniana przy użyciu hasła udostępnionego przez klienta.

Podsumowując: Identyfikacja klienta to wybór tożsamości klienta. W zależności od kontekstu klient jest identyfikowany przez ClientIdentifier(identyfikator klienta), Username(Nazwa użytkownika), wspólną tożsamość klienta utworzoną przez administratora lub certyfikat klienta. Identyfikator klienta używany do sprawdzania autentyczności nie musi być tym samym identyfikatorem, który jest używany do autoryzacji.

Programy klienckie MQTT ustawiają wartości Nazwa użytkownika i Hasło , które są wysyłane do serwera przy użyciu kanału MQTT. Mogą także ustawiać właściwości protokołu SSL, które są wymagane do szyfrowania i uwierzytelniania połączenia. Administrator decyduje o tym, czy uwierzytelnić kanał MQTT, a także określić sposób uwierzytelniania kanału.

Aby autoryzować klienta MQTT w celu uzyskania dostępu do obiektów WebSphere MQ , autoryzuj ClientIdentifierlub Username klienta albo autoryzuj wspólną tożsamość klienta. Aby umożliwić klientowi nawiązanie połączenia z produktem WebSphere MQ, należy uwierzytelnić nazwę użytkownika Nazwa użytkownikalub użyć certyfikatu klienta. Skonfiguruj usługę JAAS w celu uwierzytelnienia nazwy użytkownikai skonfiguruj protokół SSL w celu uwierzytelnienia certyfikatu klienta.

Jeśli w kliencie zostanie ustawione Hasło , zaszyfruj połączenie za pomocą sieci VPN lub skonfiguruj kanał MQTT tak, aby używał protokołu SSL, aby zachować hasło prywatne.

Zarządzanie certyfikatami klientów jest trudne. Z tego powodu, jeśli ryzyko związane z uwierzytelnianiem hasłem jest akceptowalne, uwierzytelnianie za pomocą hasła jest często używane do uwierzytelniania klientów.

Jeśli istnieje bezpieczny sposób zarządzania i przechowywania certyfikatu klienta, możliwe jest oparcie się na uwierzytelnianiu certyfikatów. Rzadko jednak zdarza się, że certyfikaty mogą być bezpiecznie zarządzane w różnych środowiskach, w których używana jest telemetria. Zamiast tego uwierzytelnianie urządzeń za pomocą certyfikatów klienta jest uzupełniane przez uwierzytelnianie haseł klientów na serwerze. Ze względu na dodatkową złożoność korzystanie z certyfikatów klientów jest ograniczone do wysoce wrażliwych aplikacji. Użycie dwóch form uwierzytelniania jest nazywane uwierzytelnianiem dwuskładniowym. Użytkownik musi znać jeden z czynników, na przykład hasło, i mieć inne, takie jak certyfikat.

W wysoce czułych aplikacjach, takich jak urządzenie chipowe i przypinane, urządzenie jest zamykane w trakcie produkcji, aby zapobiec ingerowaniu w wewnętrzny sprzęt i oprogramowanie. Zaufany, ograniczony czas, certyfikat klienta jest kopiowany do urządzenia. Urządzenie jest wdrażane w miejscu, w którym ma być używane. Dalsze uwierzytelnianie jest wykonywane za każdym razem, gdy urządzenie jest używane, albo przy użyciu hasła, albo na innym certyfikacie z karty Smart Card.

Tożsamość i autoryzacja klienta MQTT

Aby uzyskać dostęp do obiektów produktu WebSphere MQ , należy użyć wartości `ClientIdentifier`, `Username`lub wspólnej tożsamości klienta.

Administrator produktu WebSphere MQ ma trzy opcje wyboru tożsamości kanału MQTT. Administrator dokonuje wyboru przy definiowaniu lub modyfikowaniu kanału MQTT używanego przez klienta. Tożsamość jest używana do autoryzowania dostępu do tematów produktu WebSphere MQ . Dostępne opcje:

1. Identyfikator klienta.
2. Tożsamość, którą administrator udostępnia dla kanału.
3. Wartość `Nazwa użytkownika` przekazana z klienta MQTT.

`Nazwa użytkownika` jest atrybutem klasy opcji MqttConnect. Musi on zostać ustawiony przed nawiążniem połączenia z usługą. Jego wartość domyślna to NULL.

Za pomocą komendy WebSphere MQ **setmqaut** można wybrać obiekty, które mają być używane przez tożsamość powiązaną z kanałem MQTT, i które działania mają być używane. Na przykład, aby autoryzować tożsamość kanału, `MQTTClient`, udostępnionego przez administratora menedżera kolejek, `QM1`:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

Autoryzowanie klientów MQTT w celu uzyskania dostępu do obiektów produktu WebSphere MQ

Aby autoryzować klienty MQTT w celu publikowania i subskrybowania obiektów WebSphere MQ , należy wykonać poniższe kroki. Kroki są zgodne z czterema alternatywnymi wzorcami kontroli dostępu.

Zanim rozpoczniesz

Klienty MQTT są autoryzowane do uzyskiwania dostępu do obiektów w produkcie WebSphere MQ przez przypisanie tożsamości podczas nawiązywania połączenia z kanałem telemetryczny. Administrator produktu WebSphere MQ konfiguruje kanał pomiarowy przy użyciu programu WebSphere MQ Explorer w celu nadania klientowi jednego z trzech typów tożsamości:

1. `ClientIdentifier`
2. `Nazwa użytkownika`
3. Nazwa przypisana przez administratora do kanału.

Niezależnie od tego, który typ jest używany, tożsamość musi być zdefiniowana w produkcie WebSphere MQ jako główna przez zainstalowaną usługę autoryzacji. Domyślna usługa autoryzacji w systemie Windows lub Linux jest nazywana menedżerem uprawnień do obiektów (Object Authority Manager-OAM). Jeśli korzystasz z OAM, tożsamość musi być zdefiniowana jako identyfikator użytkownika.

Użyj tożsamości, aby nadać klientowi lub kolekcję klientów uprawnienia do publikowania lub subskrybowania tematów zdefiniowanych w produkcie WebSphere MQ. Jeśli klient MQTT zasubskrybował dany temat, należy użyć tożsamości, aby nadać mu uprawnienia do otrzymywania wynikowych publikacji.

Trudno jest zarządzać systemem z kilkudziesięcioma tysiącami klientów MQTT, z których każdy wymaga indywidualnych uprawnień dostępu. Jednym z rozwiązań jest zdefiniowanie wspólnych tożsamości i powiązanie poszczególnych klientów MQTT z jedną ze wspólnych tożsamości. Zdefiniuj tyle wspólnych tożsamości, ile potrzeba, aby zdefiniować różne kombinacje uprawnień. Innym rozwiązaniem jest napisanie własnej usługi autoryzacji, która może być łatwiejsza w obsłudze z tysiącami użytkowników niż system operacyjny.

Klientów MQTT można łączyć ze wspólnymi tożsamościami na dwa sposoby, korzystając z OAM:

1. Zdefiniuj wiele kanałów pomiarowych, z których każdy ma inny identyfikator użytkownika, który administrator przydziela za pomocą programu WebSphere MQ Explorer. Klienty łączące się z różnymi numerami portów TCP/IP są powiązane z różnymi kanałami telemetryczne i przypisują różne tożsamości.

2. Zdefiniuj pojedynczy kanał pomiarowy, ale dla każdego klienta wybierz `Nazwę użytkownika` z niewielkiego zestawu identyfikatorów użytkowników. Administrator konfiguruje kanał pomiarowy w celu wybrania klienta `Nazwa użytkownika` jako jego tożsamości.

W tym zadaniu tożsamość kanału telemetrycznego jest nazywana *mqttUser*, niezależnie od tego, jak została ustawiona. Jeśli kolekcje klientów korzystają z różnych tożsamości, należy użyć wielu *mqttUsers*, po jednej dla każdej kolekcji klientów. Ponieważ zadanie korzysta z usługi OAM, każdy *mqttUser* musi być identyfikatorem użytkownika.

O tym zadaniu

W ramach tego zadania użytkownik ma do wyboru cztery wzorce kontroli dostępu, które można dostosować do konkretnych wymagań. Wzorce różnią się granulacją kontroli dostępu.

- "Brak kontroli dostępu" na stronie 130
- "Kontrola dostępu gruboziarnistego" na stronie 130
- "Średnioziarnista kontrola dostępu" na stronie 130
- "Precyzyjna kontrola dostępu" na stronie 131

Wynikiem tych modeli jest przypisanie do produktu *mqttUsers* zestawów uprawnień do publikowania i subskrybowania produktu WebSphere MQ, a także do otrzymywania publikacji z produktu WebSphere MQ.

Brak kontroli dostępu
Klienty MQTT są nadawane z uprawnieniami administracyjnymi produktu WebSphere MQ i mogą wykonywać dowolne działania na dowolnym obiekcie.

Procedura

1. Utwórz identyfikator użytkownika *mqttUser* , który będzie działał jako tożsamość wszystkich klientów MQTT.
2. Dodaj program *mqttUser* do grupy `mqm` . Patrz sekcja Dodawanie użytkownika do grupy w systemie Windowslub Dodawanie użytkownika do grupy w systemie Linux .

Kontrola dostępu gruboziarnistego
Klienty MQTT mają uprawnienia do publikowania i subskrybowania oraz wysyłania komunikatów do klientów MQTT. Nie mają uprawnień do wykonywania innych działań lub uzyskiwania dostępu do innych obiektów.

Procedura

1. Utwórz identyfikator użytkownika *mqttUser* , który będzie działał jako tożsamość wszystkich klientów MQTT.
2. Autoryzowanie produktu *mqttUser* do publikowania i subskrybowania wszystkich tematów oraz wysyłanie publikacji do klientów MQTT.

```
setmqaut -m qMgr -t topic -n SYSTEM.BASE.TOPIC -p mqttUser -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqttUser -all +put
```

Średnioziarnista kontrola dostępu
Klienty MQTT są podzielone na różne grupy, aby publikować i subskrybować różne zestawy tematów, a także wysyłać komunikaty do klientów MQTT.

Procedura

1. W drzewie tematów publikowania/subskrypcji utwórz wiele identyfikatorów użytkowników, *mqttUsers*i wiele tematów administracyjnych.
2. Autoryzuj różne *mqttUsers* do różnych tematów.

```
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. Utwórz grupę *mqtt*i dodaj wszystkie *mqttUsers* do grupy.
4. Autoryzuj *mqtt* do wysyłania tematów do klientów MQTT.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Precyzyjna kontrola dostępu
Klienty MQTT są wbudowane w istniejący system kontroli dostępu, który autoryzuje grupy do wykonywania działań na obiektach.

O tym zadaniu

Identyfikator użytkownika jest przypisany do jednej lub większej liczby grup systemów operacyjnych, w zależności od wymaganych przez niego autoryzacji. Jeśli aplikacje produktu WebSphere MQ publikują i subskrybują ten sam obszar tematu co klienty MQTT, należy użyć tego modelu. Grupy są określane jako Publish*X*, Subscribe*Y*i mqtt .

Publish*X*
Członkowie grup produktu Publish*X* mogą publikować w produkcie *topicX*.

Subscribe*Y*
Członkowie grup produktu Subscribe*Y* mogą subskrybować produkt *topicY*.

mqtt
Członkowie grupy *mqtt* mogą wysyłać publikacje do klientów MQTT.

Procedura

1. Utwórz wiele grup, Publish*X* i Subscribe*Y* , które są przydzielane do wielu tematów administracyjnych w drzewie tematów publikowania/subskrypcji.
2. Utwórz grupę mqtt.
3. Utwórz wiele identyfikatorów użytkowników, *mqttUsers*i dodaj użytkowników do dowolnej z grup, w zależności od tego, do czego są uprawnieni.
4. Autoryzuj różne grupy produktów Publish*X* i Subscribe*X* do różnych tematów, a następnie autoryzuj grupę *mqtt* do wysyłania komunikatów do klientów MQTT.

```
setmqaut -m qMgr -t topic -n topic1 -p PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Uwierzytelnianie klienta MQTT przy użyciu hasła

Uwierzytelniaj nazwę użytkownika przy użyciu hasła klienta. Istnieje możliwość uwierzytelnienia klienta przy użyciu innej tożsamości do tożsamości używanej do autoryzowania klienta do publikowania i subskrybowania tematów.

Usługa telemetryczna (MQXR) korzysta z usługi JAAS w celu uwierzytelnienia klienta Nazwa użytkownika. Usługa JAAS korzysta z hasła dostarczonego przez klienta MQTT.

Administrator produktu WebSphere MQ podejmuje decyzję, czy uwierzytelnić nazwę użytkownika, czy nie w ogóle uwierzytelnić, konfigurując kanał połączenia MQTT, z którym łączy się klient. Klienci mogą być przypisani do różnych kanałów, a każdy kanał może być skonfigurowany do uwierzytelniania swoich klientów na różne sposoby. Za pomocą usługi JAASmożna skonfigurować, które metody muszą uwierzytelniać klienta, i które opcjonalnie mogą uwierzytelnić klienta.

Wybór tożsamości do uwierzytelniania nie ma wpływu na wybór tożsamości do autoryzacji. Użytkownik może chcieć skonfigurować wspólną tożsamość w celu autoryzacji dla wygody administratora, ale uwierzytelnia każdego użytkownika w celu użycia tej tożsamości. Poniższa procedura przedstawia kroki w celu uwierzytelnienia poszczególnych użytkowników w celu użycia wspólnej tożsamości:

1. Administrator produktu WebSphere MQ ustawia tożsamość kanału MQTT na dowolną nazwę, taką jak `MQTTClientUser`, za pomocą programu WebSphere MQ Explorer.
2. Administrator produktu WebSphere MQ autoryzuje produkt `MQTTClient` do publikowania i subskrybowania dowolnego tematu:

```
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. Programista aplikacji klienckiej MQTT tworzy obiekt `MqttConnectOptions` i ustawia wartości `Username` i `Password` przed nawiązywaniem połączenia z serwerem.
4. Programista zabezpieczeń tworzy usługę JAAS `LoginModule` w celu uwierzytelnienia `nazwy użytkownika` przy użyciu `hasła` i dołącza ją do pliku konfiguracyjnego JAAS .
5. Administrator produktu WebSphere MQ konfiguruje kanał MQTT w celu uwierzytelnienia użytkownika `UserName` klienta przy użyciu usługi JAAS.

Uwierzytelnianie klienta MQTT przy użyciu protokołu SSL

Połączenia między klientem MQTT a menedżerem kolejek są zawsze inicjowane przez klient MQTT. Klient MQTT jest zawsze klientem SSL. Uwierzytelnianie klienta serwera i uwierzytelnianie serwera klienta MQTT są opcjonalne.

Dzięki udostępnieniu klienta z prywatnym podpisanym certyfikatem cyfrowym można uwierzytelnić klienta MQTT w produkcie IBM WebSphere MQ. Administrator produktu IBM WebSphere MQ może wymusić na klientach MQTT uwierzytelnianie się w menedżerze kolejek przy użyciu protokołu SSL. Tylko w ramach uwierzytelniania wzajemnego można zażądać uwierzytelnienia klienta.

Alternatywę dla protokołu SSL stanowią niektóre rodzaje wirtualnej sieci prywatnej (VPN), takie jak IPsec, uwierzytelniające punkty końcowe połączenia TCP/IP. Sieć VPN szyfruje każdy pakiet IP przepływający przez sieć. Po ustanowieniu połączenia VPN uzyskana sieć jest zaufana. Klienty MQTT można połączyć z kanałami pomiarowymi przy użyciu połączeń TCP/IP za pośrednictwem sieci VPN.

Uwierzytelnianie klienta przy użyciu protokołu SSL wymaga, aby klient miał klucz tajny. Klucz tajny to klucz prywatny klienta stosowany na potrzeby certyfikatu samopodpisanego lub klucz udostępniony przez ośrodek certyfikacji. Ten klucz służy do podpisywania certyfikatu cyfrowego klienta. Odpowiedni klucz publiczny umożliwia weryfikację certyfikatu cyfrowego. Certyfikaty mogą być zaufanymi certyfikatami. Jeśli są połączone w łańcuch, można prześledzić łańcuch certyfikatów aż do zaufanego certyfikatu głównego. Podczas weryfikacji klienta wszystkie certyfikaty tworzące łańcuch certyfikatów udostępnione przez klienta są wysyłane na serwer. Serwer sprawdza łańcuch certyfikatów, aż znajdzie zaufany certyfikat. Zaufany certyfikat to certyfikat publiczny wygenerowany na podstawie certyfikatu samopodpisanego lub certyfikat główny wydany (zwykle) przez ośrodek certyfikacji. W ramach opcjonalnego ostatniego kroku zaufany certyfikat można porównać z aktualną listą odwołań certyfikatów.

Zaufanym certyfikatem może być certyfikat wystawiony przez ośrodek certyfikacji, który został już uwzględniony w bazie certyfikatów środowiska JRE. Może to być certyfikat samopodpisany lub dowolny certyfikat, który został dodany do magazynu kluczy kanału pomiarowego jako zaufany certyfikat.

Uwaga: Kanał pomiarowy ma połączony magazyn kluczy / magazyn zaufanych certyfikatów, w którym przechowywane są klucze prywatne do jednego lub większej liczby kanałów pomiarowych oraz wszystkie certyfikaty publiczne wymagane na potrzeby uwierzytelniania klientów. Baza certyfikatów środowiska JRE nigdy nie jest przywoływana, ponieważ kanał SSL musi mieć magazyn kluczy i jest to ten sam plik, który jest używany na potrzeby magazynu zaufanych certyfikatów kanału. Oznacza to, że jeśli uwierzytelnianie klienta wymaga certyfikatu głównego ośrodka CA, należy umieścić certyfikat główny w magazynie kluczy tego kanału nawet wtedy, gdy główny certyfikat ośrodka CA znajduje się już w bazie certyfikatów środowiska JRE. Baza certyfikatów środowiska JRE nigdy nie jest przywoływana.

Należy pomyśleć o zagrożeniach, jakim ma przeciwdziałać uwierzytelnianie klienta, oraz o rolach klienta i serwera w procesie przeciwdziałania tym zagrożeniom. Uwierzytelnienie samego certyfikatu klienta nie blokuje w pełni możliwości dostępu do systemu bez uprawnień. Jeśli inna osoba wejdzie w posiadanie urządzenia z klientem, urządzenie z klientem nie będzie działało z upoważnienia właściciela certyfikatu. Nigdy nie należy polegać na jednej formie obrony przed niepożądanymi atakami. Należy stosować przynajmniej uwierzytelnianie dwuskładnikowe i oprócz posiadania certyfikatu wymagać znajomości

prywatnych informacji. Na przykład można zastosować usługę JAAS i uwierzytelniać klient przy użyciu hasła wystawionego przez serwer.

Dostanie się w niepowołane ręce stanowi główne zagrożenie dla certyfikatu klienta. Certyfikat jest przechowywany w kliencie w magazynie kluczy zabezpieczonym hasłem. W jaki sposób jest on umieszczany w magazynie kluczy? W jaki sposób klient MQTT uzyskuje hasło do magazynu kluczy? Jak wysoki poziom bezpieczeństwa zapewnia zabezpieczenie hasłem? Często urządzenie pomiarowe można łatwo odłączyć. W takiej sytuacji haker może włamać się do urządzenia w miejscu prywatnym. Czy sprzęt urządzenia musi być zabezpieczony przed manipulacją? Uznaje się, że rozpowszechnianie i ochrona certyfikatów po stronie klienta to trudne zadanie – jest to tzw. problem z zarządzaniem kluczami.

Dodatkowe zagrożenie stanowi nieprawidłowe używanie urządzenia w celu uzyskiwania dostępu do serwerów w niezamierzony sposób. Na przykład zmanipulowanie aplikacji MQTT umożliwia wykorzystanie luki w konfiguracji serwera przy użyciu uwierzytelnionej tożsamości klienta.

Aby móc uwierzytelniać klienta MQTT przy użyciu protokołu SSL, należy skonfigurować kanał pomiarowy i klient.

-
-

Konfiguracja kanału telemetrycznego dla uwierzytelniania klienta MQTT przy użyciu protokołu SSL

Administrator produktu IBM WebSphere MQ konfiguruje kanały telemetryczne na serwerze. Każdy kanał jest konfigurowany tak, aby akceptował połączenie TCP/IP na innym porcie. Kanały SSL są skonfigurowane w taki sposób, że dostęp do plików kluczy jest chroniony hasłem. Jeśli kanał SSL jest zdefiniowany bez hasła lub pliku kluczy, kanał nie zaakceptuje połączeń SSL.

Ustaw właściwość `com.ibm.mq.MQTT.ClientAuth` kanału pomiarowego SSL na `REQUIRED` , aby wymusić na wszystkich klientach łączących się na tym kanale, aby dostarczyć dowód na to, że zweryfikowano certyfikaty cyfrowe. Certyfikaty klienta są uwierzytelniane przy użyciu certyfikatów z ośrodków certyfikacji, co prowadzi do zaufanego certyfikatu głównego. Jeśli certyfikat klienta jest samopodpisany lub podpisany przez certyfikat, który pochodzi z ośrodka certyfikacji, publicznie podpisane certyfikaty klienta lub ośrodka certyfikacji muszą być przechowywane w sposób bezpieczny na serwerze.

Umieść publicznie podpisany certyfikat klienta lub certyfikat z ośrodka certyfikacji w magazynie kluczy kanału telemetrycznego. Na serwerze podpisane publicznie certyfikaty są przechowywane w tym samym pliku kluczy, co certyfikaty podpisane prywatnie, a nie w osobnym magazynie zaufanych certyfikatów.

Serwer weryfikuje podpisy wszystkich certyfikatów klientów, które są wysyłane przy użyciu wszystkich certyfikatów publicznych i zestawów algorytmów szyfrowania, które ma. Serwer weryfikuje łańcuch kluczy. Menedżer kolejek można skonfigurować w taki sposób, aby testował certyfikat na liście odwołań certyfikatów. Właściwość listy nazw odwołań do menedżera kolejek to `SSLCRLNL`.

Jeśli którekolwiek z certyfikatów wysyła klient jest weryfikowane przez certyfikat w magazynie kluczy serwera, klient zostanie uwierzytelniony.

Administrator produktu WebSphere MQ może skonfigurować ten sam kanał pomiarowy, aby używać usługi JAAS w celu sprawdzenia wartości `UserName` lub `ClientIdentifier` klienta przy użyciu klienta `Password`.

Tego samego magazynu kluczy można użyć dla wielu kanałów pomiarowych.

Weryfikacja co najmniej jednego certyfikatu cyfrowego w magazynie kluczy klienta zabezpieczonego hasłem na urządzeniu uwierzytelnia klienta na serwerze. Certyfikat cyfrowy jest używany tylko do uwierzytelniania w produkcie WebSphere MQ. Nie jest on używany do weryfikowania adresu TCP/IP klienta ani do ustawiania tożsamości klienta w celu autoryzacji lub rozliczania. Tożsamość klienta adoptowane przez serwer to `Nazwa użytkownika` lub `ClientIdentifier` klienta lub tożsamość utworzona przez administratora produktu WebSphere MQ .

Do uwierzytelniania klientów można również używać zestawów algorytmów szyfrowania SSL. Poniżej znajduje się alfabetyczna lista obsługiwanych obecnie zestawów algorytmów szyfrowania SSL:

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V 7.5.0.2 Jeśli planowane jest użycie zestawów algorytmów szyfrowania produktu SHA-2 , należy zapoznać się z Wymagania systemowe dotyczące korzystania z zestawów algorytmów szyfrowania produktu SHA-2 z kanałami MQTT.

Pojęcia pokrewne

“Konfiguracja kanału telemetrycznego dla uwierzytelniania kanału za pomocą protokołu SSL” na stronie 135

Administrator produktu IBM WebSphere MQ konfiguruje kanały telemetryczne na serwerze. Każdy kanał jest konfigurowany tak, aby akceptował połączenie TCP/IP na innym porcie. Kanały SSL są skonfigurowane w taki sposób, że dostęp do plików kluczy jest chroniony hasłem. Jeśli kanał SSL jest zdefiniowany bez hasła lub pliku kluczy, kanał nie zaakceptuje połączeń SSL.

CipherSpecs i CipherSuites

Odsyłacze pokrewne

ZDEFINIUJ KANAŁ (MQTT)

ZMIEŃ KANAŁ (MQTT)

Uwierzytelnianie kanału telemetrycznego przy użyciu protokołu SSL

Połączenia między klientem MQTT a menedżerem kolejek są zawsze inicjowane przez klient MQTT. Klient MQTT jest zawsze klientem SSL. Uwierzytelnianie klienta serwera i uwierzytelnianie serwera klienta MQTT są opcjonalne.

Klient zawsze będzie próbował uwierzytelnić serwer, chyba że klient został skonfigurowany pod kątem używania specyfikacji szyfrowania obsługującej połączenia anonimowe. Jeśli uwierzytelnianie nie powiedzie się, połączenie nie zostanie nawiązane.

Alternatywę dla protokołu SSL stanowią niektóre rodzaje wirtualnej sieci prywatnej (VPN), takie jak IPsec, uwierzytelniające punkty końcowe połączenia TCP/IP. Sieć VPN szyfruje każdy pakiet IP przepływający przez sieć. Po ustanowieniu połączenia VPN uzyskana sieć jest zaufana. Klienty MQTT można połączyć z kanałami pomiarowymi przy użyciu połączeń TCP/IP za pośrednictwem sieci VPN.

Uwierzytelnianie serwera za pomocą protokołu SSL uwierzytelnia serwer, do którego użytkownik chce wysłać informacje poufne. Klient wykonuje sprawdzenia zgodne z certyfikatami wysłanym z serwera, z certyfikatami umieszczonym w magazynie zaufanych certyfikatów lub w jego składnicy JRE `cacerts` .

Składnica certyfikatów środowiska JRE to plik JKS `cacerts`. Znajduje się on w katalogu JRE `InstallPath\lib\security\`. Jest on instalowany z domyślnym hasłem `changeit`. Zaufane certyfikaty mogą być przechowywane w bazie certyfikatów środowiska JRE lub w magazynie zaufanych certyfikatów klienta. Nie można używać obu składnic. Należy skorzystać z magazynu zaufanych certyfikatów, jeśli certyfikaty publiczne i zaufane certyfikaty klienta mają być przechowywane w innym miejscu niż certyfikaty używane przez inne aplikacje Java. Należy skorzystać z bazy certyfikatów środowiska JRE, jeśli wszystkie aplikacje Java działające na kliencie mają używać wspólnej bazy certyfikatów. W przypadku wybrania bazy certyfikatów środowiska JRE należy przejrzeć certyfikaty w niej zawarte, aby upewnić się, że są to zaufane certyfikaty.

Konfigurację JSSE można zmodyfikować, podając innego dostawcę zaufania. Istnieje możliwość dostosowania dostawcy zaufania tak, aby wykonywał inne operacje sprawdzania certyfikatu. W niektórych środowiskach OGSi, które korzystały z klienta MQTT, środowisko udostępnia innego dostawcę zaufania.

Aby uwierzytelnić kanał pomiarowy przy użyciu protokołu SSL, należy skonfigurować serwer i klient.

-
-

Konfiguracja kanału telemetrycznego dla uwierzytelniania kanału za pomocą protokołu SSL

Administrator produktu IBM WebSphere MQ konfiguruje kanały telemetryczne na serwerze. Każdy kanał jest konfigurowany tak, aby akceptował połączenie TCP/IP na innym porcie. Kanały SSL są skonfigurowane

w taki sposób, że dostęp do plików kluczy jest chroniony hasłem. Jeśli kanał SSL jest zdefiniowany bez hasła lub pliku kluczy, kanał nie zaakceptuje połączeń SSL.

Zapisz certyfikat cyfrowy serwera, podpisany za pomocą jego klucza prywatnego, w magazynie kluczy, który ma być używany przez kanał pomiarowy na serwerze. Jeśli chcesz przesłać łańcuch kluczy do klienta, zapisz wszystkie certyfikaty w jego łańcuchu kluczy w magazynie kluczy. Skonfiguruj kanał pomiarowy przy użyciu programu WebSphere MQ Explorer w celu użycia protokołu SSL. Podaj ścieżkę do magazynu kluczy i frazę hasła, aby uzyskać dostęp do magazynu kluczy. Jeśli numer portu TCP/IP kanału nie zostanie ustawiony, wartością domyślną numeru portu kanału pomiarowego SSL jest 8883.

Do uwierzytelniania kanału można również używać zestawów algorytmów szyfrowania SSL. Poniżej znajduje się alfabetyczna lista obsługiwanych obecnie zestawów algorytmów szyfrowania SSL:

- 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

- V 7.5.0.2 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 Jeśli planowane jest użycie zestawów algorytmów szyfrowania produktu SHA-2 , należy zapoznać się z Wymagania systemowe dotyczące korzystania z zestawów algorytmów szyfrowania produktu SHA-2 z kanałami MQTT.

Pojęcia pokrewne

“Konfiguracja kanału telemetrycznego dla uwierzytelniania klienta MQTT przy użyciu protokołu SSL” na stronie 133

Administrator produktu IBM WebSphere MQ konfiguruje kanały telemetryczne na serwerze. Każdy kanał jest konfigurowany tak, aby akceptował połączenie TCP/IP na innym porcie. Kanały SSL są skonfigurowane w taki sposób, że dostęp do plików kluczy jest chroniony hasłem. Jeśli kanał SSL jest zdefiniowany bez hasła lub pliku kluczy, kanał nie zaakceptuje połączeń SSL.

CipherSpecs i CipherSuites

Odsyłacze pokrewne

ZDEFINIUJ KANAŁ (MQTT)

ZMIEŃ KANAŁ (MQTT)

Prywatność publikacji na kanałach telemetrycznych

Prywatność publikacji MQTT wysyłanych w obu kierunkach w kanałach telemetrycznych jest zabezpieczona za pomocą protokołu SSL w celu szyfrowania transmisji przez połączenie.

Klienty MQTT łączające się z kanałami telemetrycznymi korzystają z protokołu SSL w celu zabezpieczenia prywatności publikacji przesyłanych w kanale za pomocą kryptografii klucza symetrycznego. Ponieważ punkty końcowe nie są uwierzytelniane, szyfrowanie kanału nie jest możliwe tylko w przypadku szyfrowania. Połączenie zabezpieczania prywatności z serwerem lub uwierzytelniania wzajemnego.

Alternatywę dla protokołu SSL stanowią niektóre rodzaje wirtualnej sieci prywatnej (VPN), takie jak IPsec, uwierzytelniające punkty końcowe połączenia TCP/IP. Sieć VPN szyfruje każdy pakiet IP przepływający przez sieć. Po ustanowieniu połączenia VPN uzyskana sieć jest zaufana. Klienty MQTT można połączyć z kanałami pomiarowymi przy użyciu połączeń TCP/IP za pośrednictwem sieci VPN.

W przypadku typowej konfiguracji, która szyfruje kanał i uwierzytelnia serwer, należy zapoznać się z “Uwierzytelnianie kanału telemetrycznego przy użyciu protokołu SSL” na stronie 135.

Szyfrowanie połączeń SSL bez uwierzytelniania serwera naraża połączenie z atakami typu man-in-the-middle. Mimo, że informacje, które wymieniasz, są chronione przed podsłuchami, nie wiesz, z kim się wymieniasz. Jeśli sieć nie jest sterowana, użytkownik jest narażony na przechwytywanie transmisji danych IP i maskowanie jako punkt końcowy.

Szyfrowane połączenie SSL można utworzyć bez uwierzytelniania serwera przy użyciu wymiany kluczy Diffie-Hellman CipherSpec , która obsługuje anonimowy protokół SSL. Główny klucz tajny, współużytkowany między klientem i serwerem i używany do szyfrowania transmisji SSL, jest ustanawiany bez wymiany certyfikatu serwera prywatnego.

Ponieważ anonimowe połączenia są niezabezpieczone, większość implementacji protokołu SSL nie jest domyślnie używana do używania anonimowych specyfikacji CipherSpecs. Jeśli żądanie klienta dla połączenia SSL jest akceptowane przez kanał pomiarowy, kanał musi mieć magazyn kluczy chroniony hasłem. Domyślnie, ponieważ implementacje protokołu SSL nie używają anonimowych specyfikacji

CipherSpecs, magazyn kluczy musi zawierać podpisany przez klienta certyfikat, który może być uwierzytelniony przez klienta.

Jeśli używana jest anonimowa opcja CipherSpecs, magazyn kluczy serwera musi istnieć, ale nie musi zawierać żadnych prywatnych certyfikatów podpisanych.

Innym sposobem nawiązania szyfrowanego połączenia jest zastąpienie dostawcy zaufanego na kliencie we własnej implementacji. Dostawca zaufania nie uwierzytelniłby certyfikatu serwera, ale połączenie będzie szyfrowane.

Konfiguracja SSL klientów MQTT i kanałów telemetrycznych

Klienty MQTT i WebSphere MQ Telemetry (MQXR) używają rozszerzenia Java Secure Socket Extension (JSSE) do łączenia kanałów pomiarowych przy użyciu protokołu SSL. Demon IBM WebSphere MQ Telemetry dla urządzeń nie obsługuje protokołu SSL.

Skonfiguruj protokół SSL w celu uwierzytelnienia kanału pomiarowego, klienta MQTT i zaszyfruj przesyłanie komunikatów między klientami a kanałem pomiarowym.

Alternatywę dla protokołu SSL stanowią niektóre rodzaje wirtualnej sieci prywatnej (VPN), takie jak IPsec, uwierzytelniające punkty końcowe połączenia TCP/IP. Sieć VPN szyfruje każdy pakiet IP przepływający przez sieć. Po ustanowieniu połączenia VPN uzyskana sieć jest zaufana. Klienty MQTT można połączyć z kanałami pomiarowymi przy użyciu połączeń TCP/IP za pośrednictwem sieci VPN.

Istnieje możliwość skonfigurowania połączenia między klientem Java MQTT a kanałem telemetryczny w celu użycia protokołu SSL przez protokół TCP/IP. To, co jest zabezpieczone, zależy od tego, jak skonfigurować protokół SSL do korzystania z rozszerzenia JSSE. Począwszy od konfiguracji najbardziej zabezpieczonej, można skonfigurować trzy różne poziomy zabezpieczeń:

1. Zezwól tylko zaufanym klientom MQTT na połączenie. Połącz klienta MQTT tylko z zaufanym kanałem telemetryczny. Szyfruj komunikaty między klientem i menedżerem kolejek; patrz “Uwierzytelnianie klienta MQTT przy użyciu protokołu SSL” na stronie 132
2. Połącz klienta MQTT tylko z zaufanym kanałem telemetryczny. Szyfruj komunikaty między klientem i menedżerem kolejek; patrz “Uwierzytelnianie kanału telemetrycznego przy użyciu protokołu SSL” na stronie 135.
3. Szyfruj komunikaty między klientem i menedżerem kolejek; patrz “Prywatność publikacji na kanałach telemetrycznych” na stronie 137.

Parametry konfiguracyjne JSSE

Zmodyfikuj parametry JSSE w taki sposób, aby zmieniły sposób konfigurowania połączenia SSL. Parametry konfiguracyjne JSSE są rozmieszczone w trzech zestawach:

1. IBM WebSphere MQ Kanał telemetryczny
2. Klient Java MQTT
3. JRE (środowisko JRE)

Skonfiguruj parametry kanału telemetrycznego przy użyciu programu IBM WebSphere MQ Explorer. Ustaw parametry klienta Java MQTT w atrybucie `MqttConnectionOptions.SSLProperties`. Zmodyfikuj parametry zabezpieczeń środowiska JRE, edytując pliki w katalogu zabezpieczeń środowiska JRE zarówno na kliencie, jak i na serwerze.

IBM WebSphere MQ kanał telemetryczny

Ustaw wszystkie parametry protokołu SSL kanału telemetrycznego przy użyciu programu WebSphere MQ Explorer.

`ChannelName`

Parametr `ChannelName` jest wymaganym parametrem we wszystkich kanałach.

Nazwa kanału identyfikuje kanał powiązany z określonym numerem portu. Nazwy kanałów, które ułatwiają administrowanie zestawami klientów MQTT.

PortNumber

Parametr PortNumber jest opcjonalnym parametrem we wszystkich kanałach. Domyślnie jest to 1883 dla kanałów TCP, a 8883 dla kanałów SSL.

Numer portu TCP/IP powiązany z tym kanałem. Klienty MQTT są połączone z kanałem przez określenie portu zdefiniowanego dla kanału. Jeśli kanał ma właściwości SSL, klient musi nawiązać połączenie przy użyciu protokołu SSL, na przykład:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileNazwa

Parametr KeyFileName (Nazwa pliku kluczy) jest wymaganym parametrem dla kanałów SSL. Należy go pominąć w przypadku kanałów TCP.

KeyFileNazwa to ścieżka do magazynu kluczy Java zawierającego certyfikaty cyfrowe, które są dostępne. Użyj JKS, JCEKS lub PKCS12 jako typu magazynu kluczy na serwerze.

Zidentyfikuj typ magazynu kluczy, używając jednego z następujących rozszerzeń:

.jks
.jceks
.p12
.pkcs12

Zakłada się, że magazyn kluczy z innym rozszerzeniem nazwy pliku jest plikiem kluczy JKS.

Istnieje możliwość połączenia jednego typu magazynu kluczy na serwerze z innymi typami magazynów kluczy na kliencie.

Umieść certyfikat prywatny serwera w magazynie kluczy. Certyfikat jest znany jako certyfikat serwera. Certyfikat może być samopodpisany lub częścią łańcucha certyfikatów, który jest podpisany przez ośrodek podpisujący.

Jeśli używany jest łańcuch certyfikatów, należy umieścić powiązane certyfikaty w magazynie kluczy serwera.

Certyfikat serwera i wszystkie certyfikaty w łańcuchu certyfikatów są wysyłane do klientów w celu uwierzytelnienia tożsamości serwera.

Jeśli parametr ClientAuth został ustawiony na wartość Required, magazyn kluczy musi zawierać wszystkie certyfikaty niezbędne do uwierzytelnienia klienta. Klient wysyła samopodpisany certyfikat lub łańcuch certyfikatów, a klient jest uwierzytelniany przez pierwszą weryfikację tego materiału względem certyfikatu w magazynie kluczy. Korzystając z łańcucha certyfikatów, jeden certyfikat może weryfikować wiele klientów, nawet jeśli są one wystawiane z różnymi certyfikatami klienta.

PassPhrase

Parametr PassPhrase jest wymaganym parametrem dla kanałów SSL. Należy go pominąć w przypadku kanałów TCP.

Fraza hasła używana jest do ochrony magazynu kluczy.

ClientAuth

Parametr ClientAuth jest opcjonalnym parametrem SSL. Domyślnie nie jest to uwierzytelnianie klienta. Należy go pominąć w przypadku kanałów TCP.

Ustaw wartość ClientAuth , jeśli usługa telemetryczna (MQXR) ma być uwierzytelniana przez klienta, zanim klient będzie mógł nawiązać połączenie z kanałem telemetryczny.

Jeśli zostanie ustawiona wartość ClientAuth, klient musi połączyć się z serwerem za pomocą protokołu SSL i uwierzytelnić serwer. W odpowiedzi na ustawienie parametru ClientAuthklient wysyła swój certyfikat cyfrowy do serwera i wszystkie inne certyfikaty w magazynie kluczy.

Jego certyfikat cyfrowy jest znany jako certyfikat klienta. Certyfikaty te są uwierzytelniane na certyfikatach znajdujących się w magazynie kluczy kanału i w sklepie JRE `cacerts` .

`CipherSuite`

Parametr `CipherSuite` jest opcjonalnym parametrem SSL. Wartością domyślną jest próba wszystkich włączonych specyfikacji CipherSpecs. Należy go pominąć w przypadku kanałów TCP.

Aby użyć konkretnej specyfikacji szyfrowania CipherSpec, należy ustawić właściwość `CipherSuite` na nazwę CipherSpec , która musi być używana do nawiązania połączenia SSL.

Usługa telemetryczna i klient MQTT negocjują wspólną specyfikację CipherSpec ze wszystkich obiektów CipherSpecs , które są włączone na każdym końcu. Jeśli określony parametr CipherSpec jest określony w obu lub obu końcach połączenia, musi być zgodny z atrybutem CipherSpec na drugim końcu.

Zainstaluj dodatkowe szyfry, dodając dodatkowych dostawców do rozszerzenia JSSE.

`Federal Information Processing Standards (FIPS)`

Tryb FIPS jest ustawieniem opcjonalnym. Domyślnie nie jest ona ustawiona.

Na panelu właściwości menedżera kolejek lub za pomocą opcji **`runmqsc`**ustaw wartość SSLFIPS. Opcja SSLFIPS określa, czy mają być używane tylko algorytmy z certyfikatem FIPS.

`Lista nazw odwołań`

Lista nazw odwołań jest ustawieniem opcjonalnym. Domyślnie nie jest ona ustawiona.

Na panelu właściwości menedżera kolejek lub za pomocą komendy **`runmqsc`**ustaw wartość SSLCRLNL. SSLCRLNL określa listę nazw obiektów informacji uwierzytelniających, które są używane do udostępniania położeń odwołań certyfikatów.

Nie są używane żadne inne parametry menedżera kolejek, które ustawiają właściwości SSL.

Klient Java MQTT

Ustaw właściwości SSL dla klienta Java w katalogu `MqttConnectionOptions.SSLProperties`, na przykład:

```
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);
```

Nazwy i wartości konkretnych właściwości są opisane w klasie `MqttConnectOptions` . W celu uzyskania odsyłaczy do dokumentacji interfejsu API klienta dla bibliotek klienta MQTT należy zapoznać się z informacjami w sekcji Skorowidz programistyczny klienta MQTT.

`Protokół`

`Protokół` jest opcjonalny.

Protokół jest wybierany w trakcie negocjacji z serwerem telemetryczny. Jeśli wymagany jest konkretny protokół, można wybrać jeden z nich. Jeśli serwer telemetryczny nie obsługuje protokołu, połączenie nie powiedzie się.

`ContextProvider`

Element `ContextProvider` jest opcjonalny.

`KeyStore`

Opcja `KeyStore` jest opcjonalna. Skonfiguruj ją, jeśli na serwerze ustawiona jest wartość `ClientAuth` , aby wymusić uwierzytelnianie klienta.

Umieść certyfikat cyfrowy klienta, podpisanego przy użyciu jego klucza prywatnego, do magazynu kluczy. Podaj ścieżkę do magazynu kluczy i hasło. Typ i dostawca są opcjonalne. Domyślnym typem dostawcy jest JKS, a dostawca IBMJCE jest dostawcą domyślnym.

Określ inny dostawca magazynu kluczy, aby odwołował się do klasy, która dodaje nowego dostawcę magazynu kluczy. Przekaz nazwę algorytmu używanego przez dostawcę magazynu kluczy do utworzenia instancji KeyManagerFactory przez ustawienie nazwy menedżera kluczy.

TrustStore

Opcja TrustStore jest opcjonalna. Użytkownik może umieścić wszystkie certyfikaty zaufane w sklepie JRE cacerts .

Skonfiguruj magazyn zaufanych certyfikatów, jeśli chcesz mieć inny magazyn zaufanych certyfikatów dla klienta. Magazyn zaufanych certyfikatów może nie zostać skonfigurowany, jeśli serwer używa certyfikatu wystawionego przez znany ośrodek CA, który ma już certyfikat główny zapisany w produkcie cacerts.

Dodaj do magazynu zaufanych certyfikatów publicznie podpisany certyfikat serwera lub certyfikat główny, a następnie podaj ścieżkę i hasło magazynu zaufanych certyfikatów. Domyślnym typem dostawcy jest JKS, a dostawca IBMJCE jest dostawcą domyślnym.

Określ inny dostawca magazynu zaufanych certyfikatów, aby odwołował się do klasy, która dodaje nowego dostawcę magazynu zaufanych certyfikatów. Przekaz nazwę algorytmu używanego przez dostawcę magazynu zaufanych certyfikatów do utworzenia instancji fabryki TrustManagerFactory przez ustawienie nazwy menedżera zaufania.

JRE (środowisko JRE)

Inne aspekty zabezpieczeń produktu Java , które mają wpływ na działanie protokołu SSL zarówno na kliencie, jak i na serwerze, są konfigurowane w środowisku JRE. Pliki konfiguracyjne w systemie Windows znajdują się w katalogu *Java Installation Directory*\jre\lib\security. Jeśli używane jest środowisko JRE dostarczane z produktem IBM WebSphere MQ , ścieżka jest wyświetlana w następującej tabeli:

Tabela 8. Ścieżki do plików przy użyciu platformy dla plików konfiguracyjnych SSL środowiska JRE

Platforma	**Ścieżka do pliku**
Windows	*WMQ Installation Directory*\java\jre\lib\security
Inne platformy UNIX and Linux	*WMQ Installation Directory*/java/jre64/jre/lib/security

Powszechnie znane uprawnienia do certyfikatów

Plik cacerts zawiera certyfikaty główne powszechnie znanych ośrodków certyfikacji. cacerts jest używany domyślnie, chyba że zostanie określony magazyn zaufanych certyfikatów. Jeśli używany jest sklep cacerts lub magazyn zaufanych certyfikatów nie jest udostępniany, należy przejrzeć i edytować listę osób podpisujących w produkcie cacerts , aby spełnić wymagania bezpieczeństwa.

Produkt cacerts można otworzyć, korzystając z komendy WebSphere MQ strmqikm, która uruchamia program narzędziowy IBM Key Management. Otwórz plik cacerts jako plik JKS, korzystając z hasła changeit. Zmodyfikuj hasło, aby zabezpieczyć plik.

Konfigurowanie klas zabezpieczeń

Użyj pliku java.security , aby zarejestrować dodatkowych dostawców zabezpieczeń i inne domyślne właściwości zabezpieczeń.

Uprawnienia

Użyj pliku java.policy , aby zmodyfikować uprawnienia nadane zasobom. javaws.policy nadaje uprawnienia do produktu javaws.jar

Siła szyfrowania

Niektóre statki JREs z obniżonym szyfrowaniem wytrzymałości. Jeśli nie można zaimportować kluczy do magazynów kluczy, przyczyną może być zredukowane szyfrowanie mocy. Należy spróbować uruchomić produkt **ikeyman** przy użyciu komendy **strmqikm** lub pobrać silne, ale

ograniczone pliki jurysdykcji z produktu Zestawy programistyczne produktu IBM , informacje o zabezpieczeniach.

Ważne: Twój kraj pochodzenia może mieć ograniczenia w imporcie, posiadaniu, użyciu lub powrotnym wywozie do innego kraju, oprogramowania szyfrującego. Przed pobraniem lub korzystaniem z nieograniczonej strategii należy sprawdzić prawa swojego kraju. Sprawdź jego regulacje, a także jego strategie dotyczące importu, posiadania, używania i ponownego eksportu oprogramowania szyfrującego, w celu ustalenia, czy jest to dozwolone.

Zmodyfikuj zaufany dostawca, aby umożliwić klientowi nawiązanie połączenia z dowolnym serwerem.

W tym przykładzie przedstawiono sposób dodawania dostawcy zaufania i odwołania go z kodu klienta MQTT. W tym przykładzie nie jest przeprowadzane uwierzytelnianie klienta lub serwera. Wynikowe połączenie SSL jest szyfrowane bez uwierzytelniania.

Fragment kodu w produkcie Rysunek 25 na stronie 142 ustawia dostawcę zaufania `AcceptAllProviders` i menedżera zaufania dla klienta MQTT.

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

Rysunek 25. Fragment kodu klienta MQTT

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
    private static final long serialVersionUID = 1L;
    public AcceptAllProvider() {
        super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
        put("TrustManagerFactory.TrustAllCertificates",
                AcceptAllTrustManagerFactory.class.getName());
    }
```

Rysunek 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() };
        }
```

Rysunek 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);
  }
}
```

Rysunek 28. AcceptAllX509TrustManager.java

Konfiguracja kanału telemetrycznego JAAS

Skonfiguruj usługę JAAS w taki sposób, aby uwierzytelniała `Nazwę użytkownika` wysłanego przez klienta.

Administrator produktu WebSphere MQ konfiguruje, które kanały MQTT wymagają uwierzytelniania klienta przy użyciu usługi JAAS. Podaj nazwę konfiguracji JAAS dla każdego kanału, który ma wykonać uwierzytelnianie JAAS . Kanały mogą korzystać z tej samej konfiguracji JAAS lub mogą używać różnych konfiguracji JAAS . Konfiguracje są definiowane w produkcie *`WMQData directory`*`\qmgrs\`*`qMgrName`*`\mqxr\jaas.config`.

Plik `jaas.config` jest organizowany przez nazwę konfiguracji JAAS . Pod każdą nazwą konfiguracji znajduje się lista konfiguracji logowania; patrz Rysunek 29 na stronie 144.

JAAS udostępnia cztery standardowe moduły logowania. Standardowe moduły logowania NT i UNIX mają ograniczoną wartość.

Moduł JndiLogin
Uwierzytelnia się przed usługą katalogową skonfigurowaną w interfejsie JNDI (Java Naming and Directory Interface).

Krb5LoginModule
Uwierzytelnia się przy użyciu protokołów Kerberos .

NTLoginModule
Uwierzytelnia się przy użyciu informacji o zabezpieczeniach systemu NT dla bieżącego użytkownika.

Moduł UnixLogin
Uwierzytelnia się przy użyciu informacji o zabezpieczeniach systemu UNIX dla bieżącego użytkownika.

Problem z używaniem produktu `NTLoginModule` lub `UnixLoginModule` polega na tym, że usługa telemetryczna (MQXR) jest uruchamiana z tożsamością `mqm` , a nie tożsamością kanału MQTT. `mqm` jest tożsamością przekazytą do produktu `NTLoginModule` lub `UnixLoginModule` w celu uwierzytelnienia, a nie tożsamości klienta.

Aby rozwiązać ten problem, napisz własny moduł logowania lub użyj innych standardowych modułów logowania. Przykładowy produkt `JAASLoginModule.java` jest dostarczany razem z produktem WebSphere MQ Telemetry. Jest to implementacja interfejsu `javax.security.auth.spi.LoginModule` . Użyj go do opracowania własnej metody uwierzytelniania.

Wszystkie nowe klasy LoginModule , które zostaną określone, muszą znajdować się w ścieżce klasy usługi telemetrii (MQXR). Nie należy umieszczać klas w katalogach produktu WebSphere MQ , które znajdują się w ścieżce klasy. Utwórz własne katalogi i zdefiniuj całą ścieżkę klasy dla usługi telemetrycznej (MQXR).

Ścieżkę klas używaną przez usługę telemetryczną (MQXR) można rozszerzyć, ustawiając ścieżkę klasy w pliku `service.env` . `CLASSPATH` musi być skapitalizowana, a instrukcja ścieżki klasy może zawierać tylko literały. W zmiennej CLASSPATH nie można używać zmiennych; na przykład wartość `CLASSPATH=%CLASSPATH%` jest niepoprawna. Usługa telemetryczna (MQXR) ustawia własną ścieżkę klasy. Zostanie do niego dodana ścieżka CLASSPATH zdefiniowana w produkcie `service.env` .

Usługa telemetryczna (MQXR) udostępnia dwa wywołania zwrotne, które zwracają wartości `Nazwa użytkownika` i `Hasło` dla klienta połączonego z kanałem MQTT. W obiekcie `MqttConnectOptions` ustawiane są wartości `Nazwa użytkownika` i `Hasło` . Zapoznaj się z przykładem dostępu do opcji `Nazwa użytkownika` i `Hasło`w sekcji Rysunek 30 na stronie 145 .

Przykłady

Przykład pliku konfiguracyjnego JAAS z jedną nazwaną konfiguracją, `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";
};
```

Rysunek 29. Przykładowy plik `jaas.config`

Przykład modułu logowania JAAS zakodowanego w taki sposób, aby otrzymywać wartości `Nazwa użytkownika` i `Hasło` udostępniane przez klienta MQTT.

```
public boolean login()
    throws javax.security.auth.login.LoginException {
  javax.security.auth.callback.Callback[] callbacks =
    new javax.security.auth.callback.Callback[2];
  callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
  callbacks[1] = new javax.security.auth.callback.PasswordCallback(
      "PasswordCallback", false);
  try {
    callbackHandler.handle(callbacks);
    String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
        .getName();
    char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
        .getPassword();
    // Accept everything.
    if (true) {
      loggedIn = true;
    } else
      throw new javax.security.auth.login.FailedLoginException("Login failed");

    principal= new JAASPrincipal(username);

  } catch (java.io.IOException exception) {
    throw new javax.security.auth.login.LoginException(exception.toString());
  } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
    throw new javax.security.auth.login.LoginException(exception.toString());
  }

  return loggedIn;
}
```

Rysunek 30. Przykładowa metoda JAASLoginModule.Login()

IBM WebSphere MQ Demon telemetryczny dla pojęć związanych z urządzeniami

Demon IBM WebSphere MQ Telemetry dla urządzeń jest zaawansowaną aplikacją kliencką MQTT V3 . Służy do przechowywania i przekazywania komunikatów z innych klientów MQTT. Łączy się on z produktem IBM WebSphere MQ , takim jak klient MQTT, ale można również połączyć z nim inne klienty MQTT.

Demon jest brokerem publikowania/subskrypcji. Klienty MQTT V3 łączą się z nim w celu publikowania i subskrybowania tematów, przy użyciu łańcuchów tematów do publikowania i filtrowania tematów w celu zasubskrybowania. Łańcuch tematu jest hierarchiczny i zawiera poziomy tematów podzielone przez produkt /. Filtry tematów są łańcuchami tematów, które mogą zawierać znaki wieloznaczne + na poziomie pojedynczego poziomu oraz wielopoziomowy znak wieloznaczny # jako ostatni fragment łańcucha tematu.

Uwaga: Znaki wieloznaczne w demonie są zgodne z bardziej restrykcyjnymi regułami produktu WebSphere Message Broker, v6. IBM WebSphere MQ jest inny. Obsługuje on wielopoziomowe znaki wieloznaczne; znaki wieloznaczne mogą znajdować się na dowolnej liczbie poziomów hierarchii, w dowolnym miejscu łańcucha tematu.

Wiele klientów MQTT v3 łączy się z demonem przy użyciu portu nasłuchiwania. Domyślny port nasłuchiwania jest modyfikowalny. Można zdefiniować wiele portów nasłuchiwania i przydzielić do nich różne przestrzenie nazw, patrz “Demon WebSphere MQ Telemetry dla portów nasłuchiwania urządzeń” na stronie 153. Demon jest sam w sobie klientem MQTT v3 . Skonfiguruj połączenie z mostem demona w celu połączenia demona z portem nasłuchiwania innego demona lub do usługi WebSphere MQ Telemetry (MQXR).

Istnieje możliwość skonfigurowania wielu mostów dla demona WebSphere MQ Telemetry dla urządzeń. Użyj mostów, aby połączyć sieć demonów, które mogą wymieniać się publikacjami.

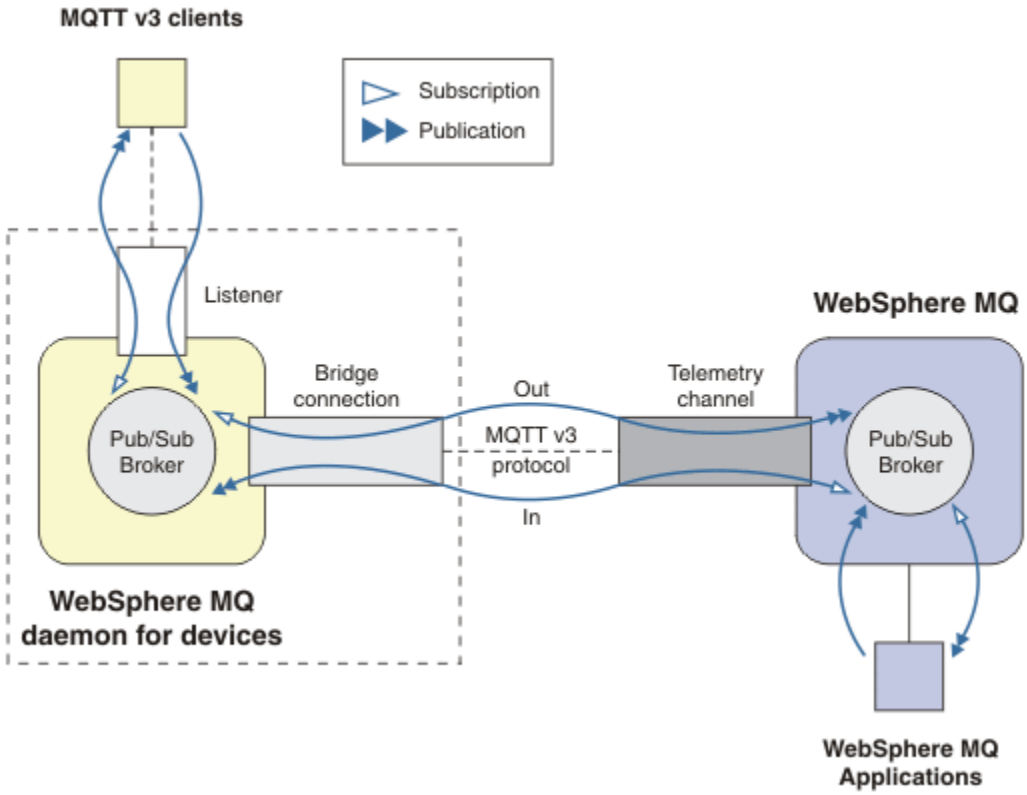
Każdy most może publikować i subskrybować tematy w swoim lokalnym demonie. Może on także publikować i subskrybować tematy w innym demonie, brokerze publikowania/subskrypcji produktu WebSphere MQ lub dowolnym innym brokerze MQTT v3 , z którym jest połączony. Korzystając z filtru tematów, można wybrać publikacje, które mają być propagowane z jednego brokera do innego. Publikacje

można propagować w dowolnym kierunku. Publikacje publiczne można propagować z demona lokalnego do każdego z przyłączonych brokerów zdalnych lub z dowolnego z przyłączonych brokerów do lokalnego demona; patrz “IBM WebSphere MQ Demon telemetryczny dla mostów urządzeń” na stronie 146.

IBM WebSphere MQ Demon telemetryczny dla mostów urządzeń

Demon IBM WebSphere MQ Telemetry dla urządzeń mostu łączy dwa brokery publikowania/subskrybowania przy użyciu protokołu MQTT v3 . Most propaguje publikacje z jednego brokera do drugiego w dowolnym kierunku. Na jednym końcu jest to demon WebSphere MQ Telemetry dla połączenia mostu urządzeń, a z drugiej strony może być menedżerem kolejek lub innym demonem. Menedżer kolejek jest połączony z połączeniem mostu za pomocą kanału telemetrycznego. Demon jest połączony z połączeniem mostu przy użyciu obiektu nasłuchiwania demona.

IBM WebSphere MQ Demon telemetryczny dla urządzeń obsługuje jedno lub większą liczbę jednoczesnych połączeń z innymi brokerami. Połączenia z demona są nazywane mostkami i są definiowane przez pozycje połączeń w pliku konfiguracyjnym demona. Połączenia z produktem IBM WebSphere MQ są nawiązywane przy użyciu kanałów pomiarowych IBM WebSphere MQ , jak pokazano na poniższym rysunku:



Rysunek 31. Łączenie IBM WebSphere MQ Telemetry daemon for devices z IBM WebSphere MQ

Most łączy demon z innym brokerem jako klient MQTT v3 . Parametry mostu odzwierciedlają atrybuty klienta MQTT v3 .

Most jest czymś więcej niż połączeniem. Działa on jako agent publikowania i subskrypcji znajdujący się między dwoma brokerami publikowania/subskrybowania. Lokalnym brokerem jest demon IBM WebSphere MQ Telemetry dla urządzeń, a broker zdalny jest dowolnym brokerem publikowania/subskrypcji, który obsługuje protokół MQTT v3 . Zwykle zdalnym brokerem jest inny demon lub IBM WebSphere MQ.

Zadaniem mostu jest propagacja publikacji pomiędzy dwoma brokerami. Most jest dwukierunkowy. Propaguje publikacje w dowolnym kierunku. Rysunek 31 na stronie 146 ilustruje sposób, w jaki most

łączy demon IBM WebSphere MQ Telemetry dla urządzeń z serwerem IBM WebSphere MQ. W produkcie “Przykładowe ustawienia tematu dla mostu” na stronie 147 użyto przykładów ilustrujących sposób użycia parametru tematu w celu skonfigurowania mostu.

Strzałki In i Out w programie Rysunek 31 na stronie 146 wskazują dwukierunkowość mostu. Na jednym końcu strzałki jest tworzona subskrypcja. Publikacje, które są zgodne z subskrypcją, są publikowane w brokerze na przeciwległy koniec strzałki. Strzałka jest oznaczona zgodnie z przepływem publikacji. Przepływ publikacji W do demona i Wyjście z demona. Znaczenie etykiet jest używane w składni komendy. Należy pamiętać, że In i Out odnoszą się do miejsca, w którym przepływ publikacji jest wysyłany, a nie do miejsca, w którym wysyłana jest subskrypcja.

Inne klienty, aplikacje lub brokery mogą być połączone z programem IBM WebSphere MQ lub z demonem WebSphere MQ Telemetry dla urządzeń. Publikują one i subskrybują tematy w brokerze, z którym są połączone. Jeśli broker ma wartość IBM WebSphere MQ, tematy mogą być grupowane lub dystrybuowane, a nie jawnie zdefiniowane w lokalnym menedżerze kolejek.

Zastosowania mostów

Połącz demonów razem przy użyciu połączeń mostów i programów nasłuchujących. Łączenie demonów i menedżerów kolejek przy użyciu połączeń mostu i kanałów telemetrycznych. Po połączeniu wielu brokerów możliwe jest utworzenie pętli. Uważaj: Publikacje mogą krążyć w nieskończoność wokół pętli brokerów, niewykryte.

Niektóre z przyczyn używania demonów mostkowanych w produkcie IBM WebSphere MQ są następujące:

Zmniejsz liczbę połączeń klienta MQTT z produktem WebSphere MQ .
Korzystając z hierarchii demonów, można połączyć wiele klientów z produktem WebSphere MQ; więcej klientów niż jeden menedżer kolejek może nawiązać połączenie jednocześnie.

Przechowywanie i przekazywanie komunikatów między klientami MQTT a produktem WebSphere MQ
Jeśli klienty nie mają własnej pamięci masowej, można użyć sklepu i dalej, aby uniknąć utrzymywania ciągłości połączeń między klientami a produktem IBM WebSphere MQ. Istnieje możliwość użycia wielu typów połączeń między klientem MQTT a produktem WebSphere MQ. Patrz sekcja Pojęcia telemetryczne i scenariusze monitorowania i sterowania.

Filtrowanie publikacji wymienianych między klientami MQTT a produktem WebSphere MQ .
Często publikacje dzielą się na komunikaty przetwarzane lokalnie i komunikaty, które obejmują inne aplikacje. Publikacje lokalne mogą obejmować przepływy sterowania między czujnikami i aktywatorami, a publikacje zdalne obejmują żądania odczytów, statusu i komend konfiguracji.

Zmiana obszarów tematów publikacji
Unikaj łańcuchów tematów z klientów przyłączonych do różnych portów nasłuchiwania przed zderzeniem się ze sobą. Przykład używa demona do oznaczania odczytów liczników pochodzących z różnych budynków. Patrz sekcja Separacja obszarów tematów różnych grup klientów.

Przykładowe ustawienia tematu dla mostu

Opublikuj wszystko w zdalnym brokerze-przy użyciu wartości domyślnych

Domyślnym kierunkiem jest out(wyjście), a most publikuje tematy w zdalnym brokerze. Parametr topic określa, które tematy są propagowane przy użyciu filtrów tematów.

Most używa parametru topic w produkcie Rysunek 32 na stronie 147 do subskrybowania wszystkiego, co zostało opublikowane w lokalnym demonie przez klienty MQTT lub przez inne brokery. Most publikuje tematy w zdalnym brokerze połączonym przez most.

```
connection Daemon1
topic #
```

Rysunek 32. Opublikuj wszystko w zdalnym brokerze

Opublikuj wszystko w zdalnym brokerze-jawne

Ustawienie topic w poniższym fragmencie kodu daje taki sam wynik, jak przy użyciu wartości domyślnych. Jedyna różnica polega na tym, że parametr **direction** jest jawny. Użyj kierunku `out` , aby zasubskrybować lokalny broker, demon i opublikować go w zdalnym brokerze. Publikacje utworzone w ramach demona lokalnego, do którego został zasubskrybowany most, są publikowane w zdalnym brokerze.

```
connection Daemon1
topic # out
```

Rysunek 33. Opublikuj wszystko w zdalnym brokerze-jawne

Opublikuj wszystko w lokalnym brokerze

Zamiast kierunku `out`, można ustawić kierunek odwrotny, `in`. Poniższy fragment kodu konfiguruje most w taki sposób, aby zasubskrybował wszystko, co zostało opublikowane w zdalnym brokerze połączonym przez most. Most publikuje tematy w lokalnym brokerze, demon.

```
connection Daemon1
topic # in
```

Rysunek 34. Opublikuj wszystko w lokalnym brokerze

Opublikuj wszystko z tematu eksportu w lokalnym brokerze do tematu importowania w brokerze zdalnym

W celu zmodyfikowania filtru tematu # w poprzednich przykładach należy użyć dwóch dodatkowych parametrów tematu, **local_prefix** i **remote_prefix**. Jeden z parametrów jest używany do modyfikowania filtru tematu używanego w subskrypcji, a drugi parametr służy do modyfikowania tematu, w którym publikowana jest publikacja. Ten efekt polega na zastąpieniu początku łańcucha tematu używanego w jednym brokerze z innym łańcuchem tematu w innym brokerze.

W zależności od kierunku komendy tematu znaczenie ma znaczenie **local_prefix** i **remote_prefix** . Jeśli kierunek to `out`, wartość domyślna **local_prefix** jest używana jako część subskrypcji tematu, a produkt **remote_prefix** zastępuje część **local_prefix** łańcucha tematu w zdalnej publikacji. If the direction is `in`, **remote_prefix** becomes part of the remote subscription, and **local_prefix** replaces the **remote_prefix** part of the topic string.

Pierwsza część łańcucha tematu jest często uznaana za definiującą obszar tematu. Aby zmienić obszar tematu, do którego jest publikowany temat, należy użyć dodatkowych parametrów. Można to zrobić, aby uniknąć propagowanego tematu kolidującego z innym tematem w brokerze docelowym lub aby usunąć łańcuch tematu punktu podłączenia.

Na przykład w poniższym fragmencie kodu wszystkie publikacje dotyczące łańcucha tematu `export/#` w demonie są ponownie publikowane w produkcie `import/#` w zdalnym brokerze.

```
topic # out export/ import/
```

Rysunek 35. Opublikuj wszystko z tematu eksportu w lokalnym brokerze do tematu importowania w brokerze zdalnym

Opublikuj wszystko w temacie importu w lokalnym brokerze z tematu eksportowania w zdalnym brokerze

Poniższy fragment kodu przedstawia zmianę konfiguracji. Most subskrybuje wszystko, co zostało opublikowane za pomocą łańcucha tematu `export/#` w zdalnym brokerze i publikuje je w produkcie `import/#` w lokalnym brokerze.

```
connection Daemon1
topic # in import/ export/
```

Rysunek 36. Opublikuj wszystko w temacie importu w lokalnym brokerze z tematu eksportowania w zdalnym brokerze

Opublikuj wszystko z punktu podłączenia 1884/ do zdalnego brokera przy użyciu oryginalnych łańcuchów tematów

W poniższym fragmencie kodu most subskrybuje wszystkie opublikowane przez klienty połączenia z punktem podłączenia 1884/ w lokalnym demonie. Most publikuje wszystko, co zostało opublikowane w punkcie podłączenia do zdalnego brokera. Łańcuch punktu podłączenia 1884/ jest usuwany z tematów opublikowanych w brokerze zdalnym. *przedrostek_lokalny* jest taki sam, jak łańcuch punktu podłączenia 1884/, a *przedrostek_zdalny* jest pustym łańcuchem.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Rysunek 37. Opublikuj wszystko z punktu podłączenia 1884/ w zdalnym brokerze przy użyciu oryginalnych łańcuchów tematów.

Oddzielanie obszarów tematów różnych klientów podłączonych do różnych demonów

Aplikacja jest napisana dla liczników energii elektrycznej, aby publikować odczyty liczników dla budynku. Odczyty są publikowane za pomocą klientów MQTT do demona udostępnianego w tym samym budynku. Temat wybrany dla publikacji to `power`. Ta sama aplikacja jest wdrażana w wielu budynkach w kompleksie. W przypadku monitorowania ośrodka i przechowywania danych odczyty ze wszystkich budynków są agregowane przy użyciu połączeń mostu. Połączenia łączą demony budynku z produktem WebSphere MQ w centralnym miejscu.

Aplikacje klienckie w każdym budynku są identyczne, ale dane muszą być zróżnicowane w zależności od budynku. Każdy odczyt ma temat `power` i musi być poprzedzony numerem budynku w celu odróżnienia go od niego. Most od pierwszego budynku w kompleksie korzysta z przedrostka `meters/building01/`, od budynku dwa przedrostek to `meters/building02/`. Odczyty z innych budynków są zgodne z tym samym wzorcem. Produkt WebSphere MQ odbiera odczyty z tematami, takimi jak `meters/building01/power`.

Przykład jest skruszony; w praktyce obszar tematu, do którego aplikacja publikuje, może być konfigurowalny.

Plik konfiguracyjny dla każdego demona zawiera instrukcję tematu, która jest zgodna z wzorcem w następującym fragmencie kodu:

```
connection Daemon1
topic power out "" meters/building01/
```

Rysunek 38. Oddziel obszary tematyczne klientów podłączonych do różnych demonów

Podaj pusty łańcuch jako obiekt zastępczy dla nieużywanego parametru local_prefix.

Oddziel obszary tematyczne klientów połączonych z tym samym demonem.

Załóżmy, że pojedynczy demon jest używany do połączenia wszystkich liczników zasilania. Zakładając, że w aplikacji można skonfigurować połączenie z różnymi portami, można odróżnić budynki, podłączając liczniki z różnych budynków do różnych portów nasłuchiwania, tak jak w poniższym fragmencie kodu. W tym przykładzie pokazano, w jaki sposób można użyć punktów podłączenia.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Rysunek 39. Oddziel obszary tematyczne klientów połączonych z tym samym demonem.

Ponownie odwzoruj różne tematy dla publikacji płynących w obu kierunkach

W konfiguracji w następującym fragmencie kodu most subskrybuje pojedynczy temat b w zdalnym brokerze i przekazuje publikacje dotyczące produktu b do demona lokalnego, zmieniając temat na a. Most subskrybuje również pojedynczy temat x w lokalnym brokerze i przekazuje publikacje dotyczące produktu x do brokera zdalnego, zmieniając temat na y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Rysunek 40. Ponownie odwzoruj różne tematy dla publikacji płynących w obu kierunkach

Ważnym punktem w tym przykładzie jest to, że różne tematy są subskrybowane i publikowane w obu brokerach. Obszary tematów w obu brokerach są rozłączne.

Ponownie odwzoruj te same tematy na publikacje, które przepływają w obu kierunkach (pętla)

W przeciwieństwie do poprzedniego przykładu, konfiguracja w produkcie Rysunek 41 na stronie 150, ogólnie, powoduje wystąpienie pętli. W instrukcji tematu `topic "" in a b`most subskrybuje produkt b zdalnie, a następnie publikuje do produktu a lokalnie. W drugiej instrukcji tematu most subskrybuje produkt a lokalnie, a następnie publikuje w produkcie b zdalnie. Tę samą konfigurację można zapisać w sposób przedstawiony w sekcji Rysunek 42 na stronie 151.

Ogólnym wynikiem jest to, że jeśli klient publikuje zdalnie produkt b , to publikacja jest przesyłana do demona lokalnego jako publikacja w temacie a. Jednak w przypadku publikowania przez most do lokalnego demona w temacie a, publikacja jest zgodna z subskrypcją udostępnianej przez most do tematu lokalnego a. Subskrypcja to `topic "" out a b`. W wyniku tego publikacja jest przesyłana z powrotem do brokera zdalnego jako publikacja w temacie b. Most jest teraz zasubskrybowany do tematu zdalnego b, a cykl rozpoczyna się ponownie.

Niektóre brokery implementują wykrywanie pętli, aby zapobiec sytuacji w pętli. Mechanizm wykrywania pętli musi działać, gdy różne typy brokery są połączone mostem. Wykrywanie pętli nie działa, jeśli produkt WebSphere MQ jest połączony z demonem produktu WebSphere MQ Telemetry dla urządzeń. Wykonuje on pracę, jeśli dwa demony produktu IBM WebSphere MQ Telemetry dla urządzeń są połączone mostem. Domyślnie funkcja wykrywania pętli jest włączona; patrz try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

*Rysunek 41. **!**Ponownie odwzoruj te same tematy na publikacje, które przepływają w obu kierunkach*

```
connection Daemon1
topic "" both a b
```

Rysunek 42. !Ponownie odwzoruj te same tematy na publikacje, które przepływają w obu kierunkach, korzystając z produktu `both`.

Konfiguracja w programie Rysunek 40 na stronie 150 jest taka sama, jak Rysunek 41 na stronie 150.

Dostępność połączeń mostu IBM WebSphere MQ Telemetry daemon for devices

Skonfiguruj wiele adresów połączeń mostu IBM WebSphere MQ Telemetry daemon for devices , aby połączyć się z pierwszym dostępnym zdalnym brokerem. Jeśli broker jest menedżerem kolejek z wieloma instancjami, podaj oba jego adresy TCP/IP. Skonfiguruj połączenie podstawowe w celu nawiązania połączenia lub ponownego nawiązania połączenia z serwerem głównym, gdy jest on dostępny.

Parametr mostu połączenia, adresy, jest listą adresów gniazd TCP/IP. Most próbuje połączyć się z każdym adresem z kolei, dopóki nie powiedzie się połączenie. Parametry połączenia round_robin i start_type sterują sposobem korzystania z adresów po pomyślnym nawiązaniu połączenia.

Jeśli parametr start_type ma wartość auto, manuallub lazy, to jeśli połączenie nie powiedzie się, most próbuje ponownie nawiązać połączenie. Wykorzystuje on każdy adres z kolei, przy czym około 20 sekund opóźnienia między każdą próbą połączenia. Jeśli parametr `start_type` ma wartość once, to jeśli połączenie nie powiedzie się, most nie podejmie próby automatycznego ponownego nawiązania połączenia.

Jeśli parametr round_robin ma wartość true, próby nawiązania połączenia z mostem rozpoczynają się od pierwszego adresu na liście, a każdy adres na liście z kolei próbuje się z nim znaleźć. Zaczyna się na pierwszym adresie ponownie, gdy lista jest wyczerpana. Jeśli na liście znajduje się tylko jeden adres, będzie on próbował to powtórzyć co 20 sekund.

Jeśli parametr round_robin ma wartość false, pierwszeństwo ma pierwszy adres na liście, który jest nazywany serwerem głównym. Jeśli pierwsza próba nawiązania połączenia z serwerem głównym nie powiedzie się, most będzie kontynuował próby ponownego nawiązania połączenia z serwerem głównym w tle. W tym samym czasie most próbuje połączyć się z innymi adresami na liście. Gdy tło próbuje nawiązać połączenie z serwerem głównym, most rozłącza się z bieżącym połączeniem i przełącza się na połączenie z serwerem podstawowym.

Jeśli połączenie zostanie odłączone dobrowolnie, na przykład za pomocą komendy **connection_stop** , po zrestartowaniu połączenia próbuje użyć tego samego adresu ponownie. Jeśli połączenie zostanie odłączone z powodu awarii połączenia lub do zdalnego brokera, który usuwa połączenie, most czeka 20 sekund. Następnie próbuje połączyć się z następnym adresem na liście lub pod tym samym adresem, jeśli na liście znajduje się tylko jeden adres.

Nawiązanie połączenia z menedżerem kolejek z wieloma instancjami

W konfiguracji menedżera kolejek z wieloma instancjami menedżer kolejek jest uruchamiany na dwóch różnych serwerach o różnych adresach IP. Zazwyczaj kanały telemetryczne są konfigurowane bez określonego adresu IP. Są one konfigurowane tylko z numerem portu. Gdy kanał pomiarowy jest uruchomiony, domyślnie wybiera pierwszy dostępny adres sieciowy na serwerze lokalnym.

Skonfiguruj parametr adresy połączenia mostu przy użyciu dwóch adresów IP używanych przez menedżer kolejek. Ustaw wartość round_robin na wartość true.

Jeśli aktywna instancja menedżera kolejek nie powiedzie się, menedżer kolejek przełącza się do instancji rezerwowej. Demon wykryje, że połączenie z aktywną instancją zostało zerwane i próbuje ponownie nawiązać połączenie z instancją rezerwową. Używa on innego adresu IP na liście adresów skonfigurowanych dla połączenia mostu.

Menedżer kolejek, z którym łączy się most, nadal jest tym samym menedżerem kolejek. Menedżer kolejek odzyskuje swój własny stan. Jeśli opcja oczyszczanie jest ustawiona na wartość false, sesja połączenia mostu jest odtwarzana do stanu sprzed przełączenia awaryjnego. Połączenie jest wznawiane po opóźnieniu. Komunikaty z "co najmniej raz" lub "co najwyżej raz" jakością usługi nie są tracone, a subskrypcje działają dalej.

Czas ponownego nawiązania połączenia zależy od liczby kanałów i klientów, które są restartowe w momencie uruchomienia instancji rezerwowej, a także od liczby komunikatów w stanie zapalnych. Połączenie mostu może próbować ponownie połączyć się z obydwoma adresami IP, ile razy przed ponownym nawiązaniem połączenia.

Nie należy konfigurować kanału telemetrycznego menedżera kolejek z wieloma instancjami przy użyciu konkretnego adresu IP. Adres IP jest poprawny tylko na jednym serwerze.

Jeśli używane jest alternatywne rozwiązanie wysokiej dostępności, które zarządza adresem IP, może być poprawne skonfigurowanie kanału telemetrycznego o określonym adresie IP.

czyszczenie (cleansession)

Połączenie mostu jest sesją klienta MQTT v3 . Można określić, czy połączenie rozpoczyna nową sesję, czy też odtwarza istniejącą sesję. Jeśli odtwarza ona istniejącą sesję, połączenie mostu zachowuje subskrypcje i zachowane publikacje z poprzedniej sesji.

Nie należy ustawiać wartości oczyszczanie na wartość false, jeśli adresy zawierają wiele adresów IP, a adresy IP łączą się z kanałami telemetrycznych udostępnianych przez różne menedżery kolejek lub do różnych demonów telemetrycznych. Stan sesji nie jest przesyłany między menedżerami kolejek lub demonami. Próba zrestartowania istniejącej sesji w innym menedżerze kolejek lub w wyniku uruchomienia nowej sesji spowoduje uruchomienie nowej sesji. Komunikaty wątpliwe są tracone, a subskrypcje mogą nie działać zgodnie z oczekiwaniami.

powiadomienia

Aplikacja może śledzić, czy połączenie mostu jest uruchomione za pomocą powiadomień. Powiadomienie jest publikacją o wartości 1, połączonej lub 0, która została rozłączona. Jest on publikowany w parametrze *topicString* zdefiniowanym przez parametr `notification_topic` . Wartością domyślną parametru *topicString* jest $SYS/`broker`/`connection`/ *clientIdentifier*/`state`. Wartość domyślna *topicString* zawiera przedrostek $SYS. Subskrybuj tematy rozpoczynające się od $SYS , definiując filtr tematów zaczynający się od $SYS. Filtr tematów #, zasubskrybuj wszystko, nie subskrybuje tematy rozpoczynające się od $SYS w demonie. Należy myśleć o $SYS , ponieważ definiując specjalny obszar tematu systemu, który różni się od obszaru tematu aplikacji.

Powiadomienia umożliwiają IBM WebSphere MQ Telemetry daemon for devices powiadamianie klientów MQTT, gdy most jest podłączony lub odłączony.

keepalive_interval

Parametr połączenia mostu keepalive_interval ustawia przedział czasu między mostem wysyłającym komendę ping TCP/IP na serwer zdalny. Domyślny okres sprawdzania wynosi 60 sekund. Komenda ping zapobiega zamknięciu sesji TCP/IP przez serwer zdalny lub przez firewall, który wykrywa okres nieaktywności w połączeniu.

CLIENTID

Połączenie mostu jest sesją klienta MQTT v3 i ma `clientIdentifier` ustawioną za pomocą parametru połączenia mostu clientid. Jeśli planowane jest wznawianie poprzedniej sesji przez ustawienie parametru cleansession na wartość false, wartość parametru `clientIdentifier` używana w każdej sesji musi być taka sama. Wartością domyślną parametru clientid jest *hostname.connectionName*, która pozostaje taka sama.

Instalowanie, weryfikowanie, konfigurowanie i sterowanie demonem produktu WebSphere MQ Telemetry dla urządzeń

Instalacja, konfiguracja i sterowanie demonem jest oparte na plikach.

Zainstaluj demon, kopiując pakiet Software Development Kit do urządzenia, na którym ma zostać uruchomiony demon .

Jako przykład uruchom program narzędziowy klienta MQTT i nawiąże połączenie z demonem WebSphere MQ Telemetry dla urządzeń jako broker publikowania/subskrypcji. Patrz sekcja Publikowanie komunikatu w konkretnym kliencie MQTT v3 .

Skonfiguruj demon, tworząc plik konfiguracyjny. Patrz sekcja WebSphere MQ Telemetry daemon for devices configuration file(Demon produktu WebSphere MQ Telemetry dla urządzeń).

Sterowanie uruchomionym demonem poprzez tworzenie komend w pliku `amqtdd.upd`. Co 5 sekund demon odczytuje ten plik, uruchamia komendy i usuwa plik. Patrz sekcja WebSphere MQ Telemetry daemon for devices command file(Demon WebSphere MQ Telemetry dla urządzeń-plik komend).

Demon WebSphere MQ Telemetry dla portów nasłuchiwania urządzeń

Połącz klienty MQTT V3 z demonem WebSphere MQ Telemetry dla urządzeń korzystających z portów nasłuchiwania. Port nasłuchiwania można kwalifikować za pomocą punktu podłączenia i maksymalnej liczby połączeń.

Port nasłuchiwania musi odpowiadać numerowi portu określonym w metodzie `connect(serverURI)` klienta MQTT klienta łączącego się z tym portem. Wartością domyślną jest zarówno klient, jak i demon 1883.

Domyślny port dla demona można zmienić, ustawiając globalną definicję `port` w pliku konfiguracyjnym demona. Konkretne porty można ustawić, dodając definicję `nasłuchiwania` do pliku konfiguracyjnego demona.

Dla każdego portu nasłuchiwania, innego niż port domyślny, można określić punkt podłączenia w celu odizolowania klientów. Klienty połączone z portem z punktem podłączenia są odizolowane od innych klientów. Patrz sekcja “Demon WebSphere MQ Telemetry dla punktów podłączenia urządzeń” na stronie 153.

Można ograniczyć liczbę klientów, które mogą łączyć się z dowolnym portem. Ustaw globalną definicję `max_connections` , aby ograniczyć połączenia do domyślnego portu, lub kwalifikuj każdy port nasłuchiwania przy użyciu parametru `max_connections`.

Przykład

Przykład pliku konfiguracyjnego, który zmienia domyślny port z 1883 na 1880i ogranicza połączenia z portem 1880 do 10000. Połączenia z portem 1884 są ograniczone do 1000. Klienty przyłączone do portu 1884 są odizolowane od klientów przyłączonych do innych portów.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Demon WebSphere MQ Telemetry dla punktów podłączenia urządzeń

Punkt podłączenia można powiązać z portem nasłuchiwania używanym przez klienty MQTT w celu nawiązania połączenia z demonem produktu WebSphere MQ Telemetry dla urządzeń. Punkt podłączenia izoluje publikacje i subskrypcje wymieniane przez klienty MQTT przy użyciu jednego portu nasłuchiwania z klientów MQTT połączonych z innym portem nasłuchiwania.

Klienty przyłączone do portu nasłuchiwania z punktem podłączenia nie mogą nigdy bezpośrednio wymieniać tematów z klientami przyłączonymi do żadnego innego portu nasłuchiwania. Klienty

przyłączone do portu nasłuchiwania bez punktu podłączenia mogą publikować lub subskrybować tematy dowolnego klienta. Klienci nie są świadomi tego, czy są one przyłączone za pośrednictwem punktu podłączenia, czy też nie. Nie ma to znaczenia dla łańcuchów tematów utworzonych przez klienty.

Punkt podłączenia jest łańcuchem tekstowym, który jest poprzedzony łańcuchem tematu publikacji i subskrypcji. Jest ona wstępna do wszystkich łańcuchów tematów utworzonych przez klienty przyłączone do portu nasłuchiwania z punktem podłączenia. Łańcuch tekstu jest usuwany ze wszystkich łańcuchów tematów wysyłanych do klientów przyłączonych do portu nasłuchiwania.

Jeśli port nasłuchiwania nie ma punktu podłączenia, łańcuchy tematów publikacji i subskrypcji utworzonych i odebranych przez klienty przyłączone do portu nie są zmieniane.

Utwórz łańcuchy punktów podłączenia na końcu `/`. W ten sposób punkt podłączenia jest tematem nadrzędnym drzewa tematów dla punktu podłączenia.

Przykład

Plik konfiguracyjny zawiera następujące porty nasłuchiwania:

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

Klient, przyłączony do portu 1883, tworzy subskrypcję programu `MyTopic`. Demon rejestruje subskrypcję jako `1883/MyTopic`. Inny klient przyłączony do portu 1883 publikuje komunikat w temacie `MyTopic`. Demon zmienia łańcuch tematu na `1883/MyTopic` i wyszukuje zgodne subskrypcje. Subskrybent na porcie 1883 odbiera publikację z oryginalnym łańcuchem tematu `MyTopic`. Demon usunł przedrostek punktu podłączenia z łańcucha tematu.

Inny klient, przyłączony do portu 1884, publikuje również w temacie `MyTopic`. Tym razem demon rejestruje temat jako `1884/MyTopic`. Subskrybent na porcie 1883 nie otrzymuje publikacji, ponieważ inny punkt podłączenia jest wynikiem subskrypcji z innym łańcuchem tematu.

Klient przyłączony do portu 1885publikuje w temacie `1883/MyTopic`informacje na temat tego tematu. Demon nie zmienia łańcucha tematu. Subskrybent na porcie 1883 otrzymuje publikację do produktu `MyTopic`.

Demon WebSphere MQ Telemetry dla urządzeń jakości usług, trwałych subskrypcji i zachowanych publikacji

Ustawienia jakości usług mają zastosowanie tylko do działającego demona. Jeśli demon zatrzyma się, niezależnie od tego, czy w sposób kontrolowany, czy z powodu awarii, stan komunikatów w stanie inflight jest tracony. Dostarczenie komunikatu co najmniej raz lub co najwyżej raz nie może być gwarantowane, jeśli demon zostanie zatrzymany. Demon WebSphere MQ Telemetry dla urządzeń obsługuje ograniczoną trwałość. Ustaw parametr konfiguracyjny **`retained_persistence`** , aby zachować zachowane publikacje i subskrypcje, gdy demon jest wyłączony.

W przeciwieństwie do produktu WebSphere MQ, demon WebSphere MQ Telemetry dla urządzeń nie kroniuje trwałych danych. Stan sesji, stan komunikatu i zachowane publikacje nie są zapisywane transakcyjnie. Domyślnie demon usuwa wszystkie dane po zatrzymaniu. Istnieje możliwość ustawienia opcji okresowo powtarzających się subskrypcji i zachowanych publikacji. Status komunikatu jest zawsze tracony, gdy demon zatrzyma się. Wszystkie publikacje niezachowane są tracone.

Ustaw opcję konfiguracyjna demona, `Retained_persistence` na `true`, aby okresowo składować przechowywane publikacje do pliku. Gdy demon zostanie zrestartowany, zachowane publikacje, które zostały ostatnio automatycznie wprowadzone, są ponownie włączone. Domyślnie zachowane komunikaty utworzone przez klienty nie są przywrócone po restarcie demona.

Ustaw opcję konfigurującą demona `Retained_persistence` na wartość `true`, aby zapisywać subskrypcje utworzone w trwałej sesji okresowo do pliku. Jeśli parametr `Retained_persistence` ma wartość `true`, subskrypcje tworzone przez klienty w sesji z ustawioną na wartość `CleanSession`

ustawioną na wartość `false`("trwała sesja") są odtwarzane. Demon odtwarza subskrypcje po restarcie, które rozpoczynają odbieranie publikacji. Klient otrzymuje publikacje po jego zrestartowaniu z produktem `CleanSession` na wartość `false`. Domyślnie stan sesji klienta nie jest zapisywany, gdy demon zatrzyma się, a więc subskrypcje nie są odtwarzane, nawet jeśli klient ustawia wartość `CleanSession` na `false`.

`Retained_persistence` -mechanizm autosave. Może nie zapisywać najnowszych zachowanych publikacji ani subskrypcji. Użytkownik może zmienić częstotliwość zapisywania zachowanych publikacji i subskrypcji. Ustaw odstęp czasu między składowaniem lub liczbę zmian między składowaniem, korzystając z opcji konfiguracyjnych `autosave_on_changes` i `autosave_interval`.

Przykładowa konfiguracja do ustawiania trwałości

```
# 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
```

Demon WebSphere MQ Telemetry dla bezpieczeństwa urządzeń

Demon WebSphere MQ Telemetry dla urządzeń może uwierzytelniać klientów, które łączą się z nim, używać referencji w celu łączenia się z innymi brokerami oraz kontrolować dostęp do tematów. Zabezpieczenia demona są ograniczone, ponieważ są one budowane przy użyciu klienta WebSphere MQ Telemetry C, który nie zapewnia obsługi SSL. W związku z tym połączenia z i z demona nie są szyfrowane i nie mogą być uwierzytelniane za pomocą certyfikatów.

Domyślnie zabezpieczenia nie są włączone.

Uwierzytelnianie klientów

Klienty MQTT mogą ustawiać nazwę użytkownika i hasło przy użyciu metod `MqttConnectOptions.setUserName` i `MqttConnectOptions.setPassword`.

Uwierzytelnij klienta, który łączy się z demonem, sprawdzając nazwę użytkownika i hasło podane przez klienta w odniesieniu do wpisów w pliku haseł. Aby włączyć uwierzytelnianie, należy utworzyć plik haseł i ustawić parametr `password_file` w pliku konfiguracyjnym demona; patrz password_file.

Ustaw parametr `allow_anonymous` w pliku konfiguracyjnym demona, aby umożliwić klientom łączenie się bez nazw użytkowników lub haseł w celu nawiązania połączenia z demonem, który sprawdza uwierzytelnianie. Patrz allow_anonymous. Jeśli klient udostępni nazwę użytkownika lub hasło, zawsze jest on sprawdzany przy użyciu pliku haseł, jeśli ustawiono parametr `password_file` .

Ustaw parametr `clientid_prefixes` w pliku konfiguracyjnym demona, aby ograniczyć połączenia do konkretnych klientów. Klienty muszą mieć wartość `clientIdentifiers` , która rozpoczyna się od jednej z przedrostków wymienionych w parametrze `clientid_prefixes` . Patrz clientid_prefixes.

Zabezpieczenia połączenia mostu

Każdy demon WebSphere MQ Telemetry dla połączenia mostu urządzeń jest klientem MQTT V3 . Nazwę użytkownika i hasło dla każdego połączenia mostu można ustawić jako parametr połączenia mostu w pliku konfiguracyjnym demona. Patrz: nazwa_użytkownika i hasło. Most może następnie uwierzytelnić się w brokerze.

Kontrola dostępu do tematów

Jeśli klienci są uwierzytelniani, demon może również zapewnić kontrolę dostępu do tematów dla każdego użytkownika. Demon nadaje kontrolę dostępu na podstawie zgodnego tematu, którego klient publikuje lub subskrybuje łańcuch tematu dostępu w pliku kontroli dostępu. Patrz acl_file.

Lista kontroli dostępu ma dwie części. Pierwsza część steruje dostępem dla wszystkich klientów, w tym klientów anonimowych. Druga część zawiera sekcję dla dowolnego użytkownika w pliku haseł. Zawiera on konkretne elementy kontroli dostępu dla każdego użytkownika.

Przykład

Parametry zabezpieczeń są przedstawione w poniższym przykładzie.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Rysunek 43. Plik konfiguracyjny demona

```
Fred:Fredpassword
Barney:Barneypassword
```

Rysunek 44. Plik haseł 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/#
```

Rysunek 45. Plik kontroli dostępu, acl.txt

Administrowanie rozgłaszanie

Informacje zawarte w tej sekcji umożliwiają zapoznanie się z zadaniami administracyjnymi WebSphere MQ Multicast, takimi jak zmniejszenie wielkości komunikatów rozsyłania grupowego i włączenie konwersji danych.

Pierwsze kroki w trybie rozsyłania grupowego

Te informacje umożliwiają rozpoczęcie pracy z tematami rozsyłania grupowego WebSphere MQ i obiektami informacji o komunikacji.

O tym zadaniu

WebSphere MQ Multicast messaging używa sieci do dostarczania komunikatów przez odwzorowanie tematów na adresy grupowe. Poniższe zadania umożliwiają szybkie przetestowanie, czy wymagany adres IP i port są poprawnie skonfigurowane do rozsyłania grupowego przesyłania komunikatów.

Tworzenie obiektu COMMINFO dla rozsyłania grupowego

Obiekt informacji o komunikacji (COMMINFO) zawiera atrybuty powiązane z transmisją rozsyłania grupowego. Więcej informacji na temat parametrów obiektu COMMINFO zawiera sekcja DEFINE COMMINFO.

Użyj następującego przykładu wiersza komend, aby zdefiniować obiekt COMMINFO dla rozsyłania grupowego:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

gdzie *MC1* jest nazwą obiektu COMMINFO, *adres grupy* jest grupowym adresem IP grupowym lub nazwą DNS, a *numer_portu* jest portem do przesłania (wartością domyślną jest 1414).

Zostanie utworzony nowy obiekt COMMINFO o nazwie *MC1* . Ta nazwa jest nazwą, którą należy określić podczas definiowania obiektu TOPIC w następnym przykładzie.

Tworzenie obiektu TOPIC na potrzeby rozsyłania grupowego

Temat jest tematem informacji, które są publikowane w komunikacie publikowania/subskrypcji, a temat jest definiowany przez utworzenie obiektu TOPIC. Obiekty TOPIC mają dwa parametry, które definiują, czy mogą być używane z rozsyłaniem grupowym, czy nie. Są to następujące parametry: **COMMINFO** i **MCAST**.

- **COMMINFO** Ten parametr określa nazwę obiektu informacji o komunikacji rozsyłania grupowego. Więcej informacji na temat parametrów obiektu COMMINFO zawiera sekcja DEFINE COMMINFO.
- **MCAST** Ten parametr określa, czy rozsyłanie grupowe jest dozwolone na tej pozycji w drzewie tematów.

Użyj następującego przykładu wiersza komend, aby zdefiniować obiekt TOPIC na potrzeby rozsyłania grupowego:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

Zostanie utworzony nowy obiekt TOPIC o nazwie *ALLSPORTS* . Ma on łańcuch tematu *Sport*, powiązany z nim obiekt informacji o komunikacji nosi nazwę *MC1* (jest to nazwa określona podczas definiowania obiektu COMMINFO w poprzednim przykładzie), a rozsyłanie grupowe jest włączone.

Testowanie publikowania/subskrypcji rozsyłania grupowego

Po utworzeniu obiektów TOPIC i COMMINFO mogą one być testowane przy użyciu przykładu `amqspubc` i przykładu `amqssubc` . Więcej informacji na temat tych przykładów można znaleźć w sekcji Programy przykładowe publikowania/subskrypcji.

1. Otwórz dwa okna wiersza komend; pierwszy wiersz komend jest przeznaczony dla przykładu publikowania produktu `amqspubc` , a drugi wiersz komend jest przeznaczony dla przykładu subskrypcji produktu `amqssubc` .
2. W wierszu komend 1 wpisz następującą komendę:

   ```
   amqspubc Sports QM1
   ```

   gdzie *Sports* to łańcuch tematu obiektu TOPIC zdefiniowany w poprzednim przykładzie, a *QM1* to nazwa menedżera kolejek.
3. W wierszu komend 2 wpisz następującą komendę:

   ```
   amqssubc Sports QM1
   ```

   gdzie *Sports* i *QM1* są takie same, jak używane w kroku “2” na stronie 157.
4. Wpisz `Hello world` w wierszu komend 1. Jeśli port i adres IP określone w obiekcie COMMINFO są poprawnie skonfigurowane; przykład `amqssubc` , który nasłuchuje na porcie dla publikacji z podanego adresu, dane wyjściowe `Hello world` w wierszu komend 2.

IBM WebSphere MQ Topologia tematu rozsyłania grupowego

W tym przykładzie przedstawiono informacje na temat topologii tematów IBM WebSphere MQ Multicast.

IBM WebSphere MQ Obsługa rozsyłania grupowego wymaga, aby każde poddrzewo miało własną grupę rozgłaszania i strumień danych w ramach ogólnej hierarchii.

Schemat adresowania IP *sieci klasowej* ma określoną przestrzeń adresową dla adresów rozsyłania. Pełny zakres rozsyłania adresu IP to adresy od `224.0.0.0` do `239.255.255.255`, ale niektóre z tych adresów są zastrzeżone. Aby uzyskać dostęp do listy adresów zastrzeżonych, należy skontaktować się

z administratorem systemu lub zapoznać się z informacjami na temat IPv4 Rejestr przestrzeni adresowej rozsyłania grupowego . Zaleca się użycie adresu rozsyłania grupowego o zasięgu lokalnym z zakresu od 239.0.0.0 do 239.255.255.255.

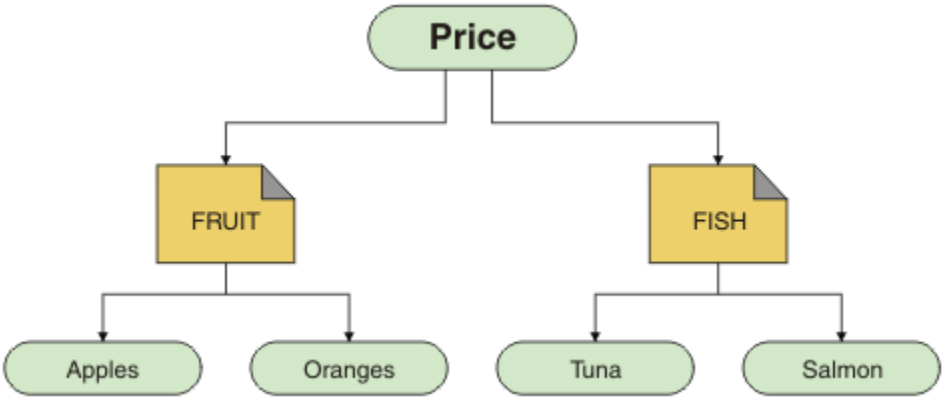
Na poniższym diagramie istnieją dwa możliwe strumienie danych rozsyłania grupowego:

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

gdzie *239.XXX.XXX.XXX* i *239.YYY.YYY.YYY* są poprawnymi adresami rozsyłania grupowego.

Te definicje tematów są używane do tworzenia drzewa tematów w sposób przedstawiony na poniższym diagramie:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)

DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Każdy obiekt informacji o komunikacji rozsyłania (multicast communication information-COMMINFO) reprezentuje inny strumień danych, ponieważ ich adresy grupowe są różne. W tym przykładzie temat FRUIT jest zdefiniowany do użycia obiektu COMMINFO MC1, temat FISH jest zdefiniowany do użycia obiektu COMMINFO MC2, a węzeł Price nie ma definicji rozsyłania grupowego.

Program WebSphere MQ Multicast ma limit 255 znaków dla łańcuchów tematów. To ograniczenie oznacza, że należy zachować ostrożność przy użyciu nazw węzłów i węzłów-liści w obrębie drzewa. Jeśli nazwy węzłów i liści są zbyt długie, łańcuch tematu może zawierać więcej niż 255 znaków i zwrócić kod przyczyny 2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR . Zaleca się, aby łańcuchy tematów były tak krótkie, jak to tylko możliwe, ponieważ dłuższe łańcuchy tematów mogą mieć szkodliwy wpływ na wydajność.

Kontrolowanie wielkości komunikatów rozsyłania grupowego

Te informacje umożliwiają zapoznanie się z formatem komunikatów produktu WebSphere MQ i zmniejszenie wielkości komunikatów produktu WebSphere MQ .

Komunikaty produktu WebSphere MQ zawierają pewną liczbę atrybutów powiązanych z tymi komunikatami, które są zawarte w deskryptorze komunikatu. W przypadku małych komunikatów atrybuty te mogą reprezentować większość ruchu danych i mogą mieć istotny szkodliwy wpływ na szybkość transmisji. WebSphere MQ Multicast enables the user to configure which, if any, of these attributes are transmitted along with the message.

Obecność atrybutów komunikatu, innych niż łańcuch tematu, zależy od tego, czy obiekt COMMINFO wskazuje, że należy je wysłać, czy nie. Jeśli atrybut nie zostanie przesłany, aplikacja odbierający zastosuje wartość domyślną. Domyślne wartości MQMD nie muszą być takie same, jak wartość MQMD_DEFAULT, i są opisane w Tabela 9 na stronie 159.

Obiekt COMMINFO zawiera atrybut MCPROP , który steruje sposobem przepływu wielu pól MQMD i właściwości użytkownika z komunikatem. Ustawiając wartość tego atrybutu na odpowiedni poziom, można kontrolować wielkość komunikatów rozsyłania grupowego produktu WebSphere MQ :

MCPROP
Właściwości rozsyłania sterują liczbą właściwości MQMD i właściwości użytkownika przepływających wraz z komunikatem.

ALL
Przesyłane są wszystkie właściwości użytkownika i wszystkie pola deskryptora MQMD.

Odpowiedź
Przesyłane są tylko właściwości użytkownika oraz pola MQMD dotyczące odpowiadania na komunikaty. Są to następujące właściwości:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- Menedżer_kolejek_zwrotnych

UŻYTKOWNIK
Przesyłane są tylko właściwości użytkownika.

BRAK
Nie są przesyłane właściwości użytkownika ani pola MQMD.

COMPAT
Ta wartość powoduje, że transmisja komunikatu jest wykonywana w trybie zgodnym z RMM, co pozwala na pewne współdziałanie z bieżącymi aplikacjami XMS i aplikacjami RMM produktu WebSphere Message Broker.

Atrybuty komunikatów rozsyłania grupowego

Atrybuty komunikatów mogą pochodzić z różnych miejsc, takich jak MQMD, pola w tabeli MQRFH2i właściwości komunikatu.

Poniższa tabela zawiera informacje o tym, co się dzieje, gdy komunikaty są wysyłane z uwzględnieniem wartości MCPROP , a także wartości domyślnej używanej, gdy atrybut nie jest wysyłany.

Tabela 9. Atrybuty przesyłania komunikatów i sposób ich powiązania z rozsyłaniem grupowym

Atrybut	**Działanie podczas korzystania z rozsyłania grupowego**	**Domyślna, jeśli nie przesłano**
TopicString	Zawsze włączone	Nie dotyczy
MQMQ StrucId	Nieprzesłane	Nie dotyczy
Wersja MQMD	Nieprzesłane	Nie dotyczy
Raport	Dołączony, jeśli nie jest domyślny	0
MsgType	Dołączony, jeśli nie jest domyślny	MQMT_DATAGRAM
Utrata ważności	Dołączony, jeśli nie jest domyślny	0
Opinie	Dołączony, jeśli nie jest domyślny	0
Kodowanie	Dołączony, jeśli nie jest domyślny	MQENC_NORMAL (equiv)
CodedCharSetId	Dołączony, jeśli nie jest domyślny	1208
Format	Dołączony, jeśli nie jest domyślny	MQRFH2
Priorytet	Dołączony, jeśli nie jest domyślny	4

Tabela 9. Atrybuty przesyłania komunikatów i sposób ich powiązania z rozsyłaniem grupowym (kontynuacja)

Atrybut	Działanie podczas korzystania z rozsyłania grupowego	Domyślna, jeśli nie przesłano
Trwałość	Dołączony, jeśli nie jest domyślny	MQPER_NOT_PERSISTENT
MsgId	Dołączony, jeśli nie jest domyślny	NULL
CorrelId	Dołączony, jeśli nie jest domyślny	NULL
BackoutCount	Dołączony, jeśli nie jest domyślny	0
ReplyToQ	Dołączony, jeśli nie jest domyślny	Wartość pusta
ReplyToQMgr	Dołączony, jeśli nie jest domyślny	Wartość pusta
UserIdentifier	Dołączony, jeśli nie jest domyślny	Wartość pusta
AccountingToken	Dołączony, jeśli nie jest domyślny	NULL
Typ IType PutApp	Dołączony, jeśli nie jest domyślny	MQAT_JAVA
Nazwa IName PutApp	Dołączony, jeśli nie jest domyślny	Wartość pusta
PutDate	Dołączony, jeśli nie jest domyślny	Wartość pusta
PutTime	Dołączony, jeśli nie jest domyślny	Wartość pusta
Dane_pochodzenia_aplikacji	Dołączony, jeśli nie jest domyślny	Wartość pusta
GroupID	Wykluczone	Nie dotyczy
Numer_kolejny_komunikatu	Wykluczone	Nie dotyczy
Depozycja	Wykluczone	Nie dotyczy
MsgFlags	Wykluczone	Nie dotyczy
OriginalLength	Wykluczone	Nie dotyczy
UserProperties	Uwzględnione	Nie dotyczy

Odsyłacze pokrewne
ALTER COMMINFO
KOMENDA DEFINE

Włączanie konwersji danych na potrzeby przesyłania komunikatów w trybie Multicast

Te informacje umożliwiają zrozumienie sposobu, w jaki konwersja danych działa w programie WebSphere MQ Multicast messaging.

WebSphere MQ Multicast to współużytkowany, bezpołączeniowy protokół, a więc nie jest możliwe, aby każdy klient wykonał konkretne żądania konwersji danych. Każdy klient zasubskrybowany w tym samym strumieniu rozsyłania grupowego odbiera te same dane binarne. Dlatego też, jeśli wymagana jest konwersja danych WebSphere MQ , konwersja jest wykonywana lokalnie na każdym kliencie.

W przypadku instalacji z różnymi platformami większość klientów wymaga danych w formacie, który nie jest rodzimą formatem aplikacji przekazującej. W takiej sytuacji wartości **CCSID** i **ENCODING** obiektu rozsyłania grupowego COMMINFO mogą zostać użyte do zdefiniowania kodowania transmisji komunikatów pod kątem wydajności.

Produkt WebSphere MQ Multicast obsługuje konwersję danych ładunku komunikatu dla następujących wbudowanych formatów:

- MQADMIN
- MQEVENT,
- MQPCF
- MQRFH,
- MQRFH2
- MQSTR

Oprócz tych formatów można również zdefiniować własne formaty i użyć wyjścia konwersji danych MQDXP-Data-conversion exit parameter .

Informacje na temat konwersji danych na temat programowania zawiera sekcja Konwersja danych w interfejsie MQI na potrzeby przesyłania komunikatów rozsyłania grupowego.

Więcej informacji na temat konwersji danych zawiera sekcja Konwersja danych.

Więcej informacji na temat wyjść konwersji danych i `ClientExitPath`zawiera sekcja ClientExitPath w pliku konfiguracyjnym klienta.

Monitorowanie aplikacji rozsyłania grupowego

Ta sekcja zawiera informacje na temat administrowania programem WebSphere MQ Multicast i monitorowania ich.

Status bieżących publikatorów i subskrybentów dla ruchu rozsyłania grupowego (na przykład liczba wysłanych i odebranych komunikatów lub liczba utraconych komunikatów) jest okresowo przesyłana do serwera od klienta. Po odebraniu statusu atrybut COMMEV obiektu COMMINFO określa, czy menedżer kolejek umieszcza komunikat zdarzenia w systemie SYSTEM.ADMIN.PUBSUB.EVENT. Komunikat o zdarzeniu zawiera odebrane informacje o statusie. Informacje te stanowią bezcenną pomoc diagnostyczną w znalezieniu źródła problemu.

Komenda MQSC **DISPLAY CONN** służy do wyświetlania informacji o połączeniu z aplikacjami połączonymi z menedżerem kolejek. Więcej informacji na temat komendy **DISPLAY CONN** zawiera sekcja DISPLAY CONN.

Aby wyświetlić status publikatorów i subskrybentów, należy użyć komendy MQSC **DISPLAY TPSTATUS** . Więcej informacji na temat komendy **DISPLAY TPSTATUS** zawiera sekcja DISPLAY TPSTATUS.

COMMEV i indykator niezawodności komunikatów rozsyłania grupowego

*Wskaźnik niezawodności*używany w połączeniu z atrybutem **COMMEV** obiektu COMMINFO jest kluczowym elementem monitorowania publikatorów i subskrybentów produktu WebSphere MQ Multicast. Indykator niezawodności (pole **MSGREL** zwracane w komendach statusu publikowania lub subskrypcji) jest indykatorem WebSphere MQ , który ilustruje procent transmisji, które nie zawierają błędów. Czasami komunikaty muszą zostać ponownie przesłane z powodu błędu transmisji, co znajduje odzwierciedlenie w wartości **MSGREL**. Potencjalne przyczyny błędów transmisji to powolne subskrybenty, sieci zajęte i wyłączenia sieci. Produkt **COMMEV** określa, czy komunikaty o zdarzeniach są generowane dla uchwytów rozsyłania grupowego, które są tworzone przy użyciu obiektu COMMINFO, i które są ustawione na jedną z trzech możliwych wartości:

WYŁĄCZONE
Komunikaty o zdarzeniach nie są zapisywane.

WŁĄCZONY
Komunikaty zdarzeń są zawsze zapisywane, z częstotliwością zdefiniowaną w parametrze COMMINFO **MONINT** .

WYJĄTEK
Komunikaty o zdarzeniach są zapisywane, gdy poziom niezawodności komunikatu jest niższy od progu niezawodności. Poziom niezawodności komunikatów wynoszący 90% lub mniej wskazuje, że może wystąpić problem z konfiguracją sieci lub że co najmniej jeden z aplikacji publikowania/subskrypcji działa zbyt wolno:

- Wartość **MSGREL(100,100)** wskazuje, że nie wystąpiły żadne problemy w krótkim czasie ani w długoterminowym przedziale czasu.
- Wartość **MSGREL(80,60)** wskazuje, że 20% komunikatów ma obecnie problemy, ale jest to również poprawa w zakresie wartości długoterminowej 60.

Klienci mogą kontynuować przesyłanie i odbieranie ruchu rozsyłania grupowego nawet wtedy, gdy połączenie pojedyncze z menedżerem kolejek zostanie zerwane, dlatego dane mogą być nieaktualne.

Niezawodność komunikatów rozsyłania grupowego

Ta sekcja zawiera informacje na temat ustawiania historii subskrypcji i komunikatów WebSphere MQ Multicast.

Kluczowym elementem pokonywania niepowodzenia transmisji przy użyciu rozsyłania grupowego jest buforowanie przesłanych danych WebSphere MQ(historia komunikatów, które mają być przechowywane w przesyłającym końcu łącza). Ten proces oznacza, że w procesie wprowadzania aplikacji nie jest wymagane buforowanie komunikatów, ponieważ produkt WebSphere MQ zapewnia niezawodność. Wielkość tej historii jest konfigurowanych za pomocą obiektu informacji o komunikacji (COMMINFO), zgodnie z opisem w poniższych informacjach. Większy bufor transmisji oznacza, że w razie potrzeby retransmitowane jest więcej historii transmisji, ale ze względu na charakter rozsyłania grupowego, nie można zapewnić 100% gwarantowanego dostarczenia.

Historia komunikatów programu WebSphere MQ Multicast jest sterowana w obiekcie informacji o komunikacji (COMMINFO) za pomocą atrybutu **MSGHIST** :

MSGHIST
Ta wartość określa wielkość historii komunikatów w kilobajtach, która jest przechowywana przez system w celu obsługi retransmisji w przypadku NACKs (ujemnych potwierdzeń).

Wartość `0` zapewnia najniższy poziom niezawodności. Wartością domyślną jest `100` kB.

Nowa historia subskrypcji produktu WebSphere MQ Multicast jest kontrolowana w obiekcie informacji o komunikacji (COMMINFO) za pomocą atrybutu **NSUBHIST** :

NSUBHIST
Historia nowego subskrybenta decyduje o tym, czy subskrybent łączący się ze strumieniem publikacji otrzymuje tyle danych, ile jest aktualnie dostępnych, czy też tylko te publikacje, które pojawiły się od czasu subskrypcji.

BRAK
Wartość `NONE` powoduje, że nadajnik przekazuje tylko publikację, która została wykonana od momentu subskrypcji. `NONE` jest wartością domyślną.

ALL
Wartość `ALL` powoduje, że nadajnik retransmituje tyle historii tematu, co jest znane. W niektórych przypadkach sytuacja taka może dać podobne zachowanie do zachowanych publikacji.

Uwaga: Użycie wartości `ALL` może mieć szkodliwy wpływ na wydajność, jeśli istnieje duża historia tematów, ponieważ cała historia tematów jest retransmitowanych.

Odsyłacze pokrewne
KOMENDA DEFINE
ALTER COMMINFO

Zaawansowane zadania rozsyłania grupowego

Te informacje umożliwiają zapoznanie się z zaawansowanymi zadaniami administracyjnymi WebSphere MQ Multicast, takimi jak konfigurowanie plików `.ini` i współdziałanie z produktem WebSphere MQ LLM.

Uwagi dotyczące zabezpieczeń w instalacji typu Multicast można znaleźć w sekcji Bezpieczeństwo w wielu rzutowaniu .

Mostkowanie między domenami publikowania/subskrybowania rozsyłania grupowego i niegrupowego

Te informacje umożliwiają zrozumienie, co się dzieje, gdy publikator inny niż rozsyłanie grupowe publikuje w temacie WebSphere MQ Multicast włączony.

Jeśli publikator inny niż rozsyłanie grupowe publikuje w temacie, który jest zdefiniowany jako **MCAST** włączony, a opcja **BRIDGE** jest włączona, menedżer kolejek przekazuje komunikat przy użyciu rozsyłania grupowego bezpośrednio do wszystkich subskrybentów, które mogą nasłuchiwać. Publikator rozsyłania grupowego nie może publikować w tematach, które nie są włączone w trybie rozsyłania grupowego.

Istniejące tematy mogą być rozsyłane grupowo, ustawiając parametry **MCAST** i **COMMINFO** dla obiektu tematu. Więcej informacji na temat tych parametrów zawiera sekcja Początkowe pojęcia dotyczące rozsyłania grupowego .

Atrybut obiektu COMMINFO **BRIDGE** steruje publikacjami z aplikacji, które nie korzystają z rozsyłania grupowego. Jeśli parametr **BRIDGE** ma wartość ENABLED , a parametr **MCAST** tematu jest również ustawiony na wartość ENABLED, publikacje z aplikacji, które nie korzystają z rozsyłania grupowego, są połączone mostem z aplikacjami, które są używane. Więcej informacji na temat parametru **BRIDGE** zawiera sekcja DEFINE COMMINFO.

Konfigurowanie plików .ini dla rozsyłania grupowego

Te informacje umożliwiają zrozumienie pól WebSphere MQ Multicast w plikach `.ini` .

Dodatkowa konfiguracja WebSphere MQ Multicast może być wykonana w pliku `ini` . Konkretny plik `ini` , który musi być używany, zależy od typu aplikacji:

- Klient: Skonfiguruj plik *MQ_DATA_PATH*`/mqclient.ini` .
- Menedżer kolejek: Skonfiguruj plik *MQ_DATA_PATH*`/qmgrs/`*QMNAME*`/qm.ini` .

gdzie *MQ_DATA_PATH* jest położeniem katalogu danych produktu WebSphere MQ (`/var/mqm/mqclient.ini`), a *QMNAME* jest nazwą menedżera kolejek, którego dotyczy plik `.ini` .

Plik `.ini` zawiera pola używane do precyzyjnego dostrajania działania WebSphere MQ Multicast:

```
Multicast:
   Protocol              = IP | UDP
   IPVersion             = IPV4 | IPV6 | ANY | BOTH
   LimitTransRate        = DISABLED | STATIC | DYNAMIC
   TransRateLimit        = 100000
   SocketTTL             = 1
   Batch                 = NO
   Loop                  = 1
   Interface             = <IPaddress>
   FeedbackMode          = ACK | NACK | WAIT1
   HeartbeatTimeout      = 20000
   HeartbeatInterval     = 2000
```

Protokół

UDP
W tym trybie pakiety są wysyłane za pomocą protokołu UDP. Elementy sieci nie mogą zapewnić asysty w dystrybucji rozsyłania grupowego, ponieważ są one jednak w trybie IP. Format pakietu pozostaje kompatybilny z PGM. Jest to wartość domyślna.

Adres IP
W tym trybie nadajnik wysyła surowe pakiety IP. Elementy sieciowe obsługiwane przez program PGM pomagają w niezawodnej dystrybucji pakietów rozsyłania grupowego. Ten tryb jest w pełni zgodny ze standardem PGM.

IPVERSION

IPV4
Komunikowanie się wyłącznie przy użyciu protokołu IPv4 . Jest to wartość domyślna.

IPV6
Komunikowanie się wyłącznie za pomocą protokołu IPv6 .

ANY
Komunikowanie się przy użyciu protokołu IPv4, IPv6lub obu tych protokołów, w zależności od tego, który protokół jest dostępny.

BOTH
Obsługuje komunikację przy użyciu protokołów IPv4 i IPv6.

Szybkość LimitTrans

WYŁĄCZONE
Nie ma kontroli szybkości transmisji. Jest to wartość domyślna.

STATYCZNA
Implementuje sterowanie statyczną szybkością transmisji. Nadajnik nie prześle w tempie przekraczającym szybkość określoną przez parametr TransRateLimit.

DYNAMICZNA
Nadajnik dostosowuje swoją szybkość transmisji zgodnie z informacją zwrotną, którą otrzymuje od odbiorników. W tym przypadku limit szybkości transmisji nie może być większy niż wartość określona za pomocą parametru Limit TransRate. Nadajnik stara się osiągnąć optymalną szybkość transmisji.

Limit TransRate
Limit szybkości transmisji danych w kb/s.

SocketTTL
Wartość parametru SocketTTL określa, czy ruch rozsyłania grupowego może przechodzić przez router, czy też liczba routerów, które może przejść przez router.

Wsadowe
Określa, czy komunikaty są wsadowe, czy wysyłane natychmiast. Istnieją dwie możliwe wartości:

- *NIE* Komunikaty nie są wsadowe, są wysyłane natychmiast.
- *TAK* Komunikaty są wsadowe.

Pętla
Aby włączyć pętlę rozsyłania grupowego, należy ustawić wartość 1 . Pętla rozsyłania grupowego definiuje, czy wysłane dane są zapętlone z powrotem do hosta, czy nie.

Interfejs
Adres IP interfejsu, na którym przepływa rozsyłanie grupowe. Więcej informacji i rozwiązywanie problemów zawiera sekcja Testowanie aplikacji rozsyłania grupowego w sieci bez rozsyłania grupowego i Ustawianie odpowiedniej sieci dla ruchu rozsyłania grupowego .

FeedbackMode

NACK
Informacje zwrotne za pomocą ujemnych potwierdzeń. Jest to wartość domyślna.

ACK
Informacje zwrotne za pomocą pozytywnych potwierdzeń.

WAIT1
Sprzężenie zwrotne za pomocą pozytywnych potwierdzeń, w których nadajnik oczekuje tylko na 1 ACK od dowolnego odbiorników.

HeartbeatTimeout
Limit czasu pulsu w milisekundach. Wartość 0 wskazuje, że zdarzenia przekroczenia limitu czasu pulsu nie są zgłaszane przez odbiornik lub odbiorniki tematu. Wartością domyślną jest 20000.

HeartbeatInterval
Odstęp czasu pulsu w milisekundach. Wartość 0 oznacza, że pulsy nie są wysyłane. Przedział czasu pulsu musi być znacznie mniejszy niż wartość **HeartbeatTimeout** , aby uniknąć fałszywych zdarzeń limitu czasu pulsu. Wartość domyślna to 2000.

Współdziałanie rozsyłania grupowego z produktem WebSphere MQ Low Latency Messaging

Te informacje umożliwiają zrozumienie współdziałania między programem WebSphere MQ Multicast i WebSphere MQ Low Latency Messaging (LLM).

Podstawowe przesyłanie ładunku jest możliwe dla aplikacji za pomocą LLM, z inną aplikacją używaniem rozsyłania grupowego w celu wymiany komunikatów w obu kierunkach. Chociaż rozsyłanie grupowe korzysta z technologii LLM, sam produkt LLM nie jest osadzony. W związku z tym możliwe jest zainstalowanie zarówno LLM, jak i WebSphere MQ Multicast, a także obsługi i serwisowania obu produktów oddzielnie.

Aplikacje LLM, które komunikują się z rozsyłaniem grupowym, mogą wymagać wysyłania i odbierania właściwości komunikatów. Właściwości komunikatu produktu WebSphere MQ i pola MQMD są przesyłane jako właściwości komunikatu LLM z określonymi kodami właściwości komunikatu LLM, jak pokazano w poniższej tabeli:

Tabela 10. Właściwości komunikatu produktu WebSphere MQ do odwzorowań właściwości LLM produktu WebSphere MQ

Właściwość produktu WebSphere MQ	**Typ właściwości LLM produktu WebSphere MQ**	**Rodzaj nieruchomości LLM**	**Kod właściwości LLM**
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	Łańcuch właściwości 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	Łańcuch właściwości LLM_PROP_KIND_String	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	Łańcuch właściwości LLM_PROP_KIND_String	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	Łańcuch właściwości LLM_PROP_KIND_String	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	Łańcuch właściwości LLM_PROP_KIND_String	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	Łańcuch właściwości LLM_PROP_KIND_String	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

Więcej informacji na temat LLM można znaleźć w dokumentacji produktu LLM: WebSphere MQ Low Latency Messaging.

administrowanieHP Integrity NonStop Server

Informacje zawarte w tej sekcji umożliwiają zapoznanie się z zadaniami administracyjnymi dla klienta IBM WebSphere MQ dla produktu HP Integrity NonStop Server.

Dostępne są dwa zadania administracyjne:

1. Samodzielne uruchamianie programu TMF/Gateway z poziomu ścieżki.
2. Zatrzymywanie programu TMF/Gateway z Pathway.

Samodzielne uruchamianie programu TMF/Gateway z ścieżki

Można zezwolić na automatyczne uruchamianie programu TMF/Gateway na pierwszym żądaniu enlistment lub można ręcznie uruchomić program TMF/Gateway z ścieżki.

Procedura

Aby ręcznie uruchomić program TMF/Gateway z ścieżki Pathway, wprowadź następującą komendę PATHCOM:

```
START SERVER <server_class_name>
```

Jeśli aplikacja kliencka wykona żądanie enlistment przed zakończeniem odtwarzania wątpliwych transakcji przez program TMF/Gateway, to żądanie jest wstrzymane do 1 sekundy. Jeśli odzyskanie nie zostanie zakończone w tym czasie, lista zostanie odrzucona. Klient otrzymuje następnie błąd MQRC_UOW_ENLISTMENT_ERROR w wyniku użycia transakcyjnego interfejsu MQI.

Zatrzymywanie programu TMF/Gateway z Pathway

W tym zadaniu opisano sposób zatrzymywania programu TMF/Gateway z ścieżki ścieżek oraz sposobu restartowania programu TMF/Gateway po jego zatrzymaniu.

Procedura

1. Aby zapobiec wprowadzeniu nowych żądań enlistment do programu TMF/Gateway, wprowadź następującą komendę:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. Aby uruchomić program TMF/Gateway w celu zakończenia wszystkich operacji w trakcie lotu i zakończyć, wprowadź następującą komendę:

   ```
   STOP SERVER <server_class_name>
   ```

3. Aby umożliwić restartowanie programu TMF/Gateway automatycznie na pierwszej liście lub ręcznie, wykonaj następujące czynności: 1 i 2, wpisz następującą komendę:

   ```
   THAW SERVER <server_class_name>
   ```

   Aplikacje mogą nie składać się z nowych żądań enlistment i nie można wydać komendy **START** do czasu wydania komendy **THAW** .

Uwagi

Niniejsza publikacja została opracowana z myślą o produktach i usługach oferowanych w Stanach Zjednoczonych.

IBM może nie oferować w innych krajach produktów, usług lub opcji omawianych w tej publikacji. Informacje o produktach i usługach dostępnych w danym kraju można uzyskać od lokalnego przedstawiciela IBM. Odwołanie do produktu, programu lub usługi IBM nie oznacza, że można użyć wyłącznie tego produktu, programu lub usługi IBM. Zamiast nich można zastosować ich odpowiednik funkcjonalny pod warunkiem, że nie narusza to praw własności intelektualnej firmy IBM. Jednakże cała odpowiedzialność za ocenę przydatności i sprawdzenie działania produktu, programu lub usługi, pochodzących od producenta innego niż IBM, spoczywa na użytkowniku.

IBM może posiadać patenty lub złożone wnioski patentowe na towary i usługi, o których mowa w niniejszej publikacji. Używanie tego dokumentu nie daje żadnych praw do tych patentów. Pisemne zapytania w sprawie licencji można przesyłać na adres:

IBM Director of Licensing
IBM Corporation North Castle Drive
Armonk, NY 10504-1785
USA

Zapytania w sprawie licencji dotyczących informacji kodowanych przy użyciu dwubajtowych zestawów znaków (DBCS) należy kierować do lokalnych działów IBM Intellectual Property Department lub zgłaszać na piśmie pod adresem:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan, Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japonia

Poniższy akapit nie obowiązuje w Wielkiej Brytanii, a także w innych krajach, w których jego treść pozostaje w sprzeczności z przepisami prawa miejscowego: INTERNATIONAL BUSINESS MACHINES CORPORATION DOSTARCZA TĘ PUBLIKACJĘ W STANIE, W JAKIM SIĘ ZNAJDUJE ("AS IS"), BEZ JAKICHKOLWIEK GWARANCJI (RĘKOJMIĘ RÓWNIEŻ WYŁĄCZA SIĘ), WYRAŹNYCH LUB DOMNIEMANYCH, A W SZCZEGÓLNOŚCI DOMNIEMANYCH GWARANCJI PRZYDATNOŚCI HANDLOWEJ, PRZYDATNOŚCI DO OKREŚLONEGO CELU ORAZ GWARANCJI, ŻE PUBLIKACJA TA NIE NARUSZA PRAW OSÓB TRZECICH. Ustawodawstwa niektórych krajów nie dopuszczają zastrzeżeń dotyczących gwarancji wyraźnych lub domniemanych w odniesieniu do pewnych transakcji; w takiej sytuacji powyższe zdanie nie ma zastosowania.

Informacje zawarte w niniejszej publikacji mogą zawierać nieścisłości techniczne lub błędy typograficzne. Informacje te są okresowo aktualizowane, a zmiany te zostaną uwzględnione w kolejnych wydaniach tej publikacji. IBM zastrzega sobie prawo do wprowadzania ulepszeń i/lub zmian w produktach i/lub programach opisanych w tej publikacji w dowolnym czasie, bez wcześniejszego powiadomienia.

Wszelkie wzmianki w tej publikacji na temat stron internetowych podmiotów innych niż IBM zostały wprowadzone wyłącznie dla wygody użytkownika i w żadnym wypadku nie stanowią zachęty do ich odwiedzania. Materiały dostępne na tych stronach nie są częścią materiałów opracowanych dla tego produktu IBM, a użytkownik korzysta z nich na własną odpowiedzialność.

IBM ma prawo do używania i rozpowszechniania informacji przysłanych przez użytkownika w dowolny sposób, jaki uzna za właściwy, bez żadnych zobowiązań wobec ich autora.

Licencjobiorcy tego programu, którzy chcieliby uzyskać informacje na temat programu w celu: (i) wdrożenia wymiany informacji między niezależnie utworzonymi programami i innymi programami (łącznie z tym opisywanym) oraz (ii) wspólnego wykorzystywania wymienianych informacji, powinni skontaktować się z:

IBM Corporation Software Interoperability Coordinator, Department 49XA 3605 Highway 52 N Rochester, MN 55901 U.S.A.

© Copyright IBM Corp. 2007, 2024

Informacje takie mogą być udostępnione, o ile spełnione zostaną odpowiednie warunki, w tym, w niektórych przypadkach, zostanie uiszczona stosowna opłata.

Licencjonowany program opisany w niniejszej publikacji oraz wszystkie inne licencjonowane materiały dostępne dla tego programu są dostarczane przez IBM na warunkach określonych w Umowie IBM z Klientem, Międzynarodowej Umowie Licencyjnej IBM na Program lub w innych podobnych umowach zawartych między IBM i użytkownikami.

Wszelkie dane dotyczące wydajności zostały zebrane w kontrolowanym środowisku. W związku z tym rezultaty uzyskane w innych środowiskach operacyjnych mogą się znacząco różnić. Niektóre pomiary mogły być dokonywane na systemach będących w fazie rozwoju i nie ma gwarancji, że pomiary wykonane na ogólnie dostępnych systemach dadzą takie same wyniki. Niektóre z pomiarów mogły być estymowane przez ekstrapolację. Rzeczywiste wyniki mogą być inne. Użytkownicy powinni we własnym zakresie sprawdzić odpowiednie dane dla ich środowiska.

Informacje dotyczące produktów producentów innych niż IBM pochodzą od dostawców tych produktów, z opublikowanych przez nich zapowiedzi lub innych powszechnie dostępnych źródeł. IBM nie testował tych produktów i nie może potwierdzić dokładności pomiarów wydajności, kompatybilności ani żadnych innych danych związanych z produktami innymi niż produkty IBM. Pytania dotyczące możliwości produktów innych podmiotów niż IBM należy kierować do dostawców tych produktów.

Wszelkie stwierdzenia dotyczące przyszłych kierunków rozwoju i zamierzeń IBM mogą zostać zmienione lub wycofane bez powiadomienia.

Publikacja ta zawiera przykładowe dane i raporty używane w codziennych operacjach działalności gospodarczej. W celu kompleksowego ich zilustrowania podane przykłady zawierają nazwiska osób prywatnych, nazwy przedsiębiorstw oraz nazwy produktów. Wszystkie te nazwy/nazwiska są fikcyjne i jakiekolwiek podobieństwo do istniejących nazw/nazwisk i adresów jest całkowicie przypadkowe.

LICENCJA W ZAKRESIE PRAW AUTORSKICH:

Niniejsza publikacja zawiera przykładowe aplikacje w kodzie źródłowym, ilustrujące techniki programowania w różnych systemach operacyjnych. Użytkownik może kopiować, modyfikować i dystrybuować te programy przykładowe w dowolnej formie bez uiszczania opłat na rzecz IBM, w celu projektowania, używania, sprzedaży lub dystrybucji aplikacji zgodnych z aplikacyjnym interfejsem programistycznym dla tego systemu operacyjnego, dla którego napisane zostały programy przykładowe. Programy przykładowe nie zostały gruntownie przetestowane. IBM nie może zatem gwarantować ani sugerować niezawodności, użyteczności i funkcjonalności tych programów.

W przypadku przeglądania niniejszych informacji w formie elektronicznej, zdjęcia i kolorowe ilustracje mogą nie być wyświetlane.

Informacje dotyczące interfejsu programistycznego

Informacje dotyczące interfejsu programistycznego, o ile są udostępniane, mają być pomocne podczas tworzenia oprogramowania aplikacji do użytku z tym programem.

Podręcznik ten zawiera informacje na temat planowanych interfejsów programistycznych, które umożliwiają klientom pisanie programów w celu uzyskania dostępu do usług IBM WebSphere MQ.

Informacje te mogą również zawierać informacje na temat diagnostyki, modyfikacji i strojenia. Tego typu informacje są udostępniane jako pomoc przy debugowaniu aplikacji.

Ważne: Informacji na temat diagnostyki, modyfikacji i strojenia nie należy używać jako interfejsu programistycznego, ponieważ może on ulec zmianie.

Znaki towarowe

IBM, logo IBM , ibm.com, są znakami towarowymi IBM Corporation, zarejestrowanymi w wielu systemach prawnych na całym świecie. Aktualna lista znaków towarowych IBM jest dostępna w serwisie WWW, w sekcji "Copyright and trademark information"(Informacje o prawach autorskich i znakach towarowych), pod adresemwww.ibm.com/legal/copytrade.shtml. Nazwy innych produktów lub usług mogą być znakami towarowymi IBM lub innych podmiotów.

Microsoft oraz Windows są znakami towarowymi Microsoft Corporation w Stanach Zjednoczonych i/lub w innych krajach.

UNIX jest zastrzeżonym znakiem towarowym The Open Group w Stanach Zjednoczonych i/lub w innych krajach.

Linux jest zastrzeżonym znakiem towarowym Linusa Torvaldsa w Stanach Zjednoczonych i/lub w innych krajach.

Ten produkt zawiera oprogramowanie opracowane przez Eclipse Project (http://www.eclipse.org/).

Java oraz wszystkie znaki towarowe i logo dotyczące języka Java są znakami towarowymi lub zastrzeżonymi znakami towarowymi Oracle i/lub przedsiębiorstw afiliowanych Oracle.

Numer pozycji:

(1P) P/N: